Applications of an expressive statistical model checking approach to the analysis of genetic circuits

Paolo Ballarini, Marie Duflot

Applications of an expressive statistical model checking approach to the analysis of genetic circuits

Paolo Ballarini[a], Marie Duflot[b]

[a] *CentraleSupelec, Chatenay-Malabry, France*
[b] *Université de Lorraine, Inria & CNRS, LORIA, Vandoeuvre-lès-Nancy, France*

Abstract

Stochastic temporal logics have demonstrated their efficiency in the analysis of discrete-state stochastic models. In this paper we consider the application of a recently introduced formalism, namely the Hybrid Automata Stochastic Language (HASL), to the analysis of biological models of genetic circuits. In particular we demonstrate the potential of HASL by focusing on two aspects: first the analysis of a genetic oscillator and then the analysis of gene expression. With respect to oscillations, we formalize a number of HASL based measures which we apply on a realistic model of a three-gene repressilator. With respect to gene expression, we consider a model with delayed stochastic dynamics, a class of systems whose dynamics includes both Markovian and non-Markovian events, and we identify a number of relevant and sophisticated measures. To assess the HASL defined measures we employ the COSMOS tool, a statistical model checker designed for HASL model checking.

Key words: model checking, stochastic methods, qualitative/quantitative analysis of biochemical systems, delayed stochastic dynamics

1. Introduction

Biological systems are regulated by complex information processing mechanisms which are at the basis of their survival and adaptation to environmental changes. Despite the continuous advancements in experimental methods, many of those mechanisms are still not well understood. The end goal of computational systems biology [39] is thus to develop formal methods for rigorously representing and effectively analyzing biological systems. Understanding what cells actually compute, how they perform computations, and eventually how such computations can be modified/engineered, are essential tasks which computational modeling aims at. In this context, the ability to "interrogate" a model by asking relevant "questions", referred to as *model checking*, is critical. Model checking approaches have proved to be an effective means for analyzing biological systems, both in the framework of non-probabilistic models [28, 16] and in that of stochastic models [43, 35, 14]. In the realm of stochastic modeling, such questions have a *quantitative* nature as they most often take into account

timing as well as likelihood of events. In this paper we consider the application of a recently introduced stochastic formalism, namely HASL, to the verification of stochastic models of biological mechanisms. HASL uses Linear Hybrid Automata (LHA) as a means for characterizing relevant dynamics of a model, and, as such, it allows one to conceive and evaluate sophisticated measures.

Our contribution. We consider the application of HASL to the verification of biological systems. Our contribution is twofold. In the first part we illustrate the application of HASL to a relevant aspect of many biological mechanisms, namely the analysis of oscillatory trends in stochastic models of biological systems. In so doing we extend the very preliminary results presented in [13] (where we showed how HASL can be used to count the average number of oscillation periods on a simple example of a 3-species oscillator) by introducing an elaborate approach based on two different oscillation related behaviors (i.e. noisy periodicity and noisy alternance). We then illustrate the proposed approach on a stochastic oscillator case-study: the so called repressilator.

In the second part, we demonstrate the effectiveness of HASL verification by developing a comprehensive case study of gene expression, a relevant biological mechanism represented by means of non-Markovian models, that cannot be analyzed by means of classical (numerical) stochastic model checking.

Paper organisation. In Section 2 we present the background material the paper is based upon, *i.e.* how systems can be modeled and analyzed using HASL formalism. Section 3 is devoted to the application of HASL to the analysis of oscillations while Section 4 illustrate the HASL-based analysis of a model of gene-expression. Finally a conclusion is given in Section 5.

2. Statistical model checking with HASL

Model checking [9] is a well established methodology introduced in the early 80s for the automatic verification of discrete-state models against temporal logic properties. The original approach was then extended to probabilistic models, and in particular to continuous-time Markov Chains (CTMCs) [54], resulting in the introduction of the Continuous Stochastic Logic (CSL)[6, 8] and subsequent extensions. The basic idea of stochastic model checking is that, given a stochastic model and a property (expressed in terms of a stochastic logic formula), algorithms (either based on numerical methods or statistical ones) are applied to approximate the probability with which the property is fulfilled by the model.

The Hybrid Automata Stochastic Language (HASL) [11] is a recently introduced formalism widening the family of model checking approaches for stochastic models. Its main characteristics are as follows: first it addresses a broad class of stochastic processes which includes but, unlike most stochastic logics, is not limited to CTMCs. Second the HASL formalism turns out to be a powerful specification language through which temporal reasoning is naturally blended with elaborate reward-based analysis. In that respect HASL unifies

the expressiveness of CSL[8] and its action-based [7], timed-automata [25, 19] and reward-based [33] extensions, in a single powerful formalism. Third HASL model checking belongs to the family of statistical model checking approaches (i.e. those that employ stochastic simulation as a means to obtain estimates of relevant properties of the considered model). More specifically HASL statistical model checking employs confidence-interval methods to estimate the expected value of random variables which may represent either a measure of probability or a generic real-valued measure.

In the following we recall the basics of the HASL formalism. First we present Discrete Event Stochastic Process (DESP), the expressive class of stochastic models to which HASL formalism can be applied. Then we describe an extended version of generalized stochastic Petri nets (GSPN), called non Markovian stochastic Petri nets (NMSPN), which is the modelling formalism used within HASL (note that the semantics of an NMSPN is a DESP). We then introduce the notion of HASL specifications, which consist of an automaton for selecting relevant executions of the DESP and obtaining information about them, and a target measure to be evaluated. We finally briefly describe the statistical model checking scheme, upon which the COSMOS tool [10] (i.e. the model checker supporting the HASL approach) is based. For a comprehensive and more formal treatment of HASL we refer the reader to [11].

2.1. Discrete Event Stochastic Process

A DESP is a stochastic process consisting of a (possibly infinite) set S of states and whose dynamic is triggered by a (finite) set E of discrete events. As mentioned beforehand, no restrictions are considered on the nature of the delay distribution associated with events, thus any distribution with non-negative support may be considered. This model is very similar to Generalized Semi Markov Processes [31]. The only difference, set aside the way to present the functions, being the introduction of "indicators" needed when synchronizing the DESP with an LHA. The automaton will indeed not have access to the actual state of the DESP but will get partial information through the values of chosen indicators.

Definition 2.1. *A DESP is a tuple* $\mathcal{D} = \langle S, \pi_0, E, Ind, enabled, target, delay, choice \rangle$ *where:*

- $\boldsymbol{S}$ *is a discrete set of states,*
- $\boldsymbol{\pi_0 \in dist(S)}$ *is the initial distribution on states,*
- $\boldsymbol{E}$ *is a set of events,*
- $\boldsymbol{Ind}$ *is a set of functions from* S *to* $\mathbb{R}$ *called state indicators (including the constant functions and the set* $Prop$ *of* state propositions*, taking values in* $\{0, 1\}$*),*
- $\boldsymbol{enabled: S \to 2^E}$ *are the enabled events in each state*
- $\boldsymbol{target: S \times E \to S}$ *is a partial function describing the state reached from state* s *on occurrence of an enabled event* $e \in enabled(s)$

- $\boldsymbol{delay: S \times E \to dist(\mathbb{R}_{\geq 0})}$ *is a partial function describing the distribution of the delay of an enabled event* $e \in enabled(s)$
- $\boldsymbol{choice: S \times 2^E \to dist(E)}$ *is a partial function describing the distribution to chose among simultaneously scheduled events*

A configuration of a DESP consists in a state, a value for the current time, and a function $E \to \mathbb{R}_{\geq 0}$, called scheduling function, that says for every enabled event when it is scheduled, with value $+\infty$ if an event is not enabled. An evolution step consists in :

- firing the first scheduled event and moving to a new state according to function $target$, and updating the current time. In case two or more events are scheduled at the same time, select one using the *choice* distribution.
- update the scheduling function by removing events no longer enabled and sampling a delay for the newly enabled ones

For the sake of saving space in this paper, we omit the formal definition of DESP semantics that can be found in [11] and give an informal description of non Markovian stochastic Petri nets, the high-level language adopted to characterize DESPs in the context of HASL model checking.

2.2. DESP in terms of non Markovian stochastic Petri nets.

According to its definition, the characterization of a DESP is a rather unpractical one, requiring an explicit listing of all of its elements (i.e. states, transitions, delay distributions, probability distributions governing concurrent events). However, several high-level formalisms commonly used for representing Markov chain models (e.g. generalized stochastic Petri nets [1], Stochastic Process Algebras [32]), can straightforwardly be adapted to represent a quite expressive class of DESPs. In the context of HASL model checking, we consider non Markovian stochastic Petri nets (NMSPNs) [17] as a high level formalism for representing DESPs. Our choice of NMSPNs has been guided in particular by the following two factors: (1) they allow for efficient path generation (due the simplicity of the *firing rule* which drives their dynamics) and (2) the distributions used to sample firing times of transitions are not restricted to (a small set containing) exponential distributions and can model non-Markovian behaviors.

We quickly recall the basics about NMSPN models pointing out the correspondence with the various parts of a DESP.

A NMSPN model (e.g. Figure 1) is a bi-partite graph consisting of two classes of nodes: *places* (represented by circles) and *transitions* (represented by bars) as well as directed edges, called *arcs*, possibly annotated by a number: its *multiplicity*. Places may contain *tokens* (e.g. representing the number of molecules of a given species) while transitions (i.e. representing the events) indicate how tokens "flow" within the net. The state of a NMSPN consists of a *marking*, *i.e.* a place-vector indicating the distribution of tokens throughout the places. Given a transition t and a place p, we say that p is an input (*resp.*

output) place of t whenever there is an arc from p to t (*resp.* from t to p). A transition is enabled whenever all of its input places contain a number of tokens greater than or equal to the multiplicity of the corresponding (input) arcs. An enabled transition may *fire* consuming tokens (in a number indicated by the multiplicity of the corresponding input arcs) from all of its input places and producing tokens (in a number indicated by the multiplicity of the corresponding output arcs) in all of its output places. Transitions can be either *timed* (denoted by empty bars, if exponential, or gray bars if non-exponential) or *immediate* (denoted by black filled-in bars). Generally speaking, transitions are characterized by: (1) a distribution which randomly determines the delay before firing it (corresponding to the DESP *delay* function); (2) a priority which *deterministically* selects, among the transitions scheduled the soonest, the one to be fired; (3) a weight, that is used in the random choice between transitions scheduled the soonest with the same highest priority (priority and weight corresponding to the DESP *choice* function). With the GSPN formalism [1], the delay of timed transitions is assumed *exponentially* distributed, whereas with NMSPN, it can be given by any distribution. Thus when a GSPN timed-transition is characterized simply by its weight $t \equiv w$ ($w \in \mathbb{R}_{\geq 0}$ indicating an $Exp(w)$ distributed delay), a NMSPN timed-transition is characterized by a triple: $t \equiv (\text{Dist-t}, \text{Dist-p}, w)$, where Dist-t indicates the type of distribution (e.g. Unif), dist-p indicates the parameters of the distribution (e.g $[\alpha, \beta]$) and $w \in \mathbb{R}_{\geq 0}$ is used to probabilistically choose between transitions occurring with equal delay[1].

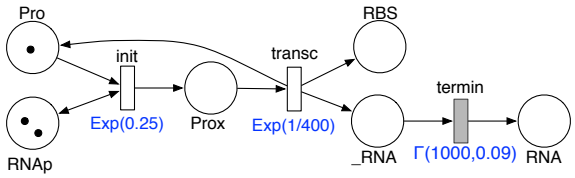


Figure 1: Example of NMSPN: model of reaction R_1 and R_2 of single-gene system

Example 2.1. *In order to illustrate the semantics of simple HASL specifications given in Example 2.2, we consider a simple model of the basic elements of the gene expression mechanism. The NMSPN presented in Figure 1 encodes two chemical reactions characterizing the transcription phase of the single-gene model described more precisely in Section 4.1. These reactions are:*

$$R_1 \quad : \quad Pro + *RNAp \xrightarrow{k_t} Prox$$
$$R_2 \quad : \quad Prox \xrightarrow{\lambda_1} Pro + RBS + RNA(\Gamma(G_{len}, 0.09))$$

If the precise description of the chemical process is of no use here, it is worth

[1] equal delay occurring with non null probability in case of non-continuous distributions

describing the reactions and the net a bit further. One place has been created for each species involved in reactions R_1 and R_2 (i.e. Pro, RNAp, Prox, RBS, RNA), plus an extra place (i.e. _RNA) for capturing the intermediate delayed phase of RNA formation. Then reaction R_1, corresponding to transition init in the net, follows a classical exponential distribution, with in this case rate $k_t = 0.25$. Reaction R_2 is a bit more complex since not all products are released at the same time. After an exponentially distributed delay of rate $\lambda_1 = 1/400$, products Pro and RBS are released (transition transc) while RNA needs another Gamma distributed delay (transition termin with parameters shape $= 1000$ and scale $= 0.09$ as from experimental data) to be released. In the initial marking $M_0 = (1, 2, 0, 0, 0, 0)$ we assume a molecule of Pro and two molecules of RNAp are available. Thus in state M_0 init is the only reaction enabled, and when it fires it will remove one token from both Pro and RNAp and add a token in each of its output places (i.e. RNAp and Prox), changing the marking into $M_1 = (0, 2, 1, 0, 0, 0)$ whereby the only enabled transition is transc. Upon firing, this transition adds a token back to place Pro which will enable again transition init and lead to a possible alternation of these two transitions.

2.3. Hybrid Automata Stochastic Language

HASL is a formalism designed to analyze properties of a DESP $\mathcal{D}$. It is quite expressive since it permits both to estimate the occurrence probability for a given set of paths and to evaluate complex measures on the system. To do so, a HASL specification is a pair $(\mathcal{A}, Z)$ where $\mathcal{A}$ is Linear Hybrid Automaton (i.e. a restriction of hybrid automata [2]) and Z is an expression, involving *data variables* of $\mathcal{A}$, that states the measure under study. The goal of HASL statistical model checking is to estimate the value of Z by synchronization of the process $\mathcal{D}$ with the automaton $\mathcal{A}$. More precisely, a timed execution of $\mathcal{D}$ is sampled following the probability distributions. By construction of the LHA, at most one execution of $\mathcal{A}$ can follow this execution of $\mathcal{D}$, and this goes on until $\mathcal{A}$ reaches some final state or the synchronization fails, which occurs when a transition of $\mathcal{D}$ cannot be followed by $\mathcal{A}$. The role of the LHA is twofold. First it permits to select paths: if a path of the corresponding DESP is relevant, the automaton reaches a final location, and if the automaton decides that a path is irrelevant (failed synchronization), it is discarded and not taken into account to compute expression Z. Second the evolution of data variables maintained by the automaton are not only used to select paths but also to get precise information about the system (such as time spent in a particular location, number of molecules of a given type, cumulated waiting time of products in a queue,...), information that will be used in the final expression Z.

2.3.1. Synchronized Linear Hybrid Automata

In order to evaluate sophisticated measures on the DESP, we need an expressive model for selecting relevant paths while computing interesting information along the paths. To do so we need more than traditional timed automata [3]. First in our model we want to consider not just clock variables but rather a set

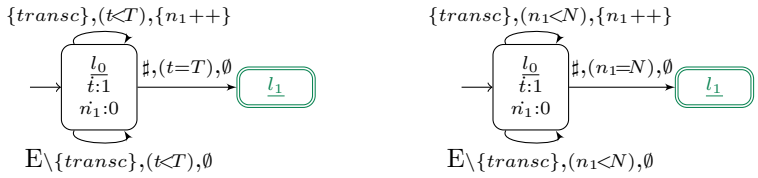
$X = (x_1, ...x_n)$ of real valued *data variables*, whose evolution rate can change depending on the location of the automaton but also on the state of the DESP. Then we also want to extend constraints (the conditions under which it is possible to fire a transition) and updates (the effect on variables of firing a transition). We thus introduce a set Const of boolean combinations of inequalities of the form $\sum_{1\leq i\leq n} \alpha_i x_i + c \prec 0$ where $\alpha_i, c \in Ind$ are indicators and $\prec \in \{=, <, >, \leq, \geq\}$, and a set Up where an update in Up is an n-tuple of functions $u_1, ..., u_n$ in which each u_k is of the form $x_k = \sum_{1\leq i\leq n} \alpha_i x_i + c$ where the α_i and c are indicators.

We then obtain a subclass of hybrid automata, enriched with indicators, that we call linear hybrid automata (LHA), which is formally defined as follows:

Definition 2.2. *A* (synchronized) LHA *is a tuple* $\mathcal{A} = \langle E, L, \Lambda, Init, Final, X, flow, \rightarrow\rangle$ *composed of:*

- $\boldsymbol{E}$*, a finite alphabet of events;*
- $\boldsymbol{L}$*, a finite set of locations;*
- $\boldsymbol{\Lambda : L \rightarrow Prop}$*, a location labeling function;*
- $\boldsymbol{Init}$*, a subset of L called the initial locations;*
- $\boldsymbol{Final}$*, a subset of L called the final locations;*
- $\boldsymbol{X = (x_1, ...x_n)}$ *a n-tuple of data variables;*
- $\boldsymbol{flow : L \mapsto Ind^n}$*, a function that associates to each location a n-tuple of indicators representing the rate of evolution of each data variable.*
- $\boldsymbol{\rightarrow \subseteq L \times 2^E \cup \{\sharp\} \times \mathsf{Const} \times \mathsf{Up} \times L}$*, the transition relation*

An edge $(l, E', \gamma, U, l') \in \rightarrow$ (also denoted $l \xrightarrow{E',\gamma,U} l'$), consists of: two locations l and l' (source and destination), either a set E' of labels telling which events of the DESP can trigger this edge of the LHA, or the label $\sharp$ denoting an autonomous (*i.e.* not synchronized) edge, a constraint $\gamma \in \mathsf{Const}$ indicating for which values of the data variables this edge can be fired, and a set U of *updates* saying how the variables' values are affected on traversing this edge. Updates and constraints are here much more expressive than those authorized in timed automata as they allow for linear combinations of data variables using indicators as coefficients. These LHA are called *synchronized* since they are meant to synchronize with a DESP through events in E and propositions in $Prop$. In the following, since all our LHA wil be synchronized, we wil just call them LHA for short. In order to ensure the uniqueness of the path of $\mathcal{A}$ corresponding to a run of the DESP, a set of further constraints are imposed on the automaton and can be found in [11]. For example, locations can be annotated with propositions, ensuring that a location can be entered only when the proposition is true (in the state of the DESP), which in particular can ensure that only one initial location is possible for each initial state of the DESP. Another condition is to restrict autonomous transitions to a set lConst of left closed constraints as they have to be taken "as soon as enabled".



(a) $\mathcal{A}_a$: time-bounded measures (b) $\mathcal{A}_b$: event-bounded measures

Figure 2: Examples of LHA for selecting paths of the NMSPN model of Figure 1

Example 2.2. *: Figure 2 depicts two variants of a simple, two locations (l_0 initial, l_1 final), LHA for selecting paths of the NMSPN toy model of Figure 1. Such LHA employs two data-variables: $t \in \mathbb{R}_{\geq 0}$, registering the simulation-time (i.e. flow $\dot{t} = 1$ in every location), and $n_1 \in \mathbb{N}$, counting the occurrences of transition transcr (i.e. flow $\dot{n_1} = 0$ in every location). The automaton synchronizes with events occurrences (i.e. firing of NMSPN transitions) and distinguishes between the occurrences of the transc event (captured by the top self-loop on l_0 through which n_1 is incremented) and the occurrences of any other event (captured by the bottom self-loop on l_0). The autonomous edge $l_0 \to l_1$ (leading to accepting location l_1) characterizes the selected paths. Thus the LHA in Figure 2(a) selects time-bounded paths corresponding to simulation time $t = T$, while the LHA in Figure 2(b) selects event-bounded paths which contains $n_1 = N$ occurrences of event transc.*

2.3.2. HASL expressions

The second component of an HASL specification is an expression, denoted Z that is either a particular operator (probability, CDF, PDF, as described later) or the expectation (E) of an arithmetic expression whose basic ingredients are data variables and that uses path operators like $last(y)$ (i.e. the last value of y along a synchronizing path, when reaching a final location), $min(y)$ ($max(y)$) the minimum (maximum), value of y along a synchronizing path), $int(y)$ (i.e. the integral over time along a path) and $avg(y)$ (the average value of y along a path).

Example 2.3. *let us consider $\mathcal{A}_a$ and $\mathcal{A}_b$ the two LHAs of Figure 2(a), respectively Figure 2(b). We provide below examples of HASL specifications obtained by combining $\mathcal{A}_a$ and $\mathcal{A}_b$ with appropriate HASL expressions:*

- $\phi_{a_1} \equiv (\mathcal{A}_a, E[last(n_1)])$*: expected number of transc events within time T.*
- $\phi_{b_1} \equiv (\mathcal{A}_b, E[last(t)])$*: expected delay for N occurrences of transc.*

Measures of probability in HASL terms. Given a set of executions Ex of the DESP, for example the set of executions satisfying a given property, one may want to compute the probability of this set of executions. Measures of probability can be obtained with HASL by building an automaton that only accepts

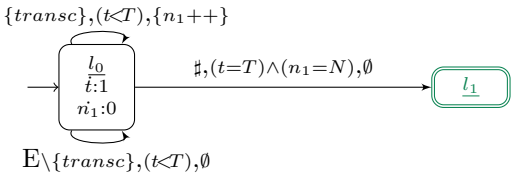


Figure 3: $\mathcal{A}_c$: detecting N occurrences of *transc* events within time T in the NMSPN model of Figure 1

executions in Ex and using a dedicated HASL expression $Z \equiv P$, the evaluation of which results in the probability that a path of the model is accepted by the LHA (this is simply given by the ratio of accepted paths over the whole set sampled from the model).

Figure 3 shows another variant of the previous LHA, which we may refer to as $\mathcal{A}_c$. In this case the selected paths are distinguished with respect to the occurrences of the *transc* event they contain: for those which contain exactly N occurrences of *transc* the variable OK is set to 1, for the remaining ones OK is set to 0. Thus an example of HASL encoded measure of probability is:

$$\phi_{c_1} \equiv (\mathcal{A}_c, P) : \text{probability of } N \text{ occurrences of } transc \text{ within time } T$$

In recent updates, the statistical model checker for HASL (introduced in next section) has been enriched with operators for automatically measuring the Probability/Cumulative Distribution Function (PDF/CDF) of the value that a variable x takes at the end of a synchronizing path. This is essentially achieved by specification of a partition of the domain of the variable x obtained through the following syntax: $Z = PDF(last(x), lower, step, stop)$ respectively $Z = CDF(last(x), lower, step, stop)$, where *lower* denotes the lower bound of the domain, *stop* the lower bound of last interval, and *step* the dimension of each sub-interval. Thus, for example, the expression $Z = PDF(last(n_1), 0, 1, 10)$ splits the domain of $n_1 \in \mathbb{N}$, into $\{0\}, \{1\} \ldots \{9\}, \{10, \ldots \infty\})$, through which the PDF of the number of occurred *transc* event within T is sampled at values 0 to 10.

2.4. The COSMOS tool: HASL statistical model checking scheme

Assessment of HASL specifications against a DESP model is performed by means of the Cosmos [21, 10] statistical model checker. Cosmos employs a classical confidence interval method [49] to obtain an estimate $\bar{Z}$ of the target expression Z. The running scheme of COSMOS is as follows: given as inputs a NMSPN model $\mathcal{D}$, an HASL specification $(\mathcal{A}, Z)$, a confidence-level ϵ and a semi-interval width δ, Cosmos applies stochastic simulation to iteratively generate trajectories of the product-process $(\mathcal{D} \times \mathcal{A})$ in an amount sufficient so that the resulting estimate $\bar{Z}$ is guaranteed to meet the required accuracy (i.e with probability $(1 - \epsilon)$ the actual value of Z will fall in the estimated interval whose semi-width is bounded by δ, i.e. $[\bar{Z} - \delta, \bar{Z} + \delta]$).

Following the characteristics of the above presented models, the synchronization of a NMSPN and an LHA admits two possible behaviors. Either it ends up in some absorbing configuration, corresponding to either a final state of the automaton or a failure of synchronization and leading to a finite synchronizing path, or the synchronization goes over all states of the path without ever reaching an absorbing configuration. In order to perform statistical verification, all the sampled paths need to be finite. In order to ensure such a property, we assume that, with probability 1, the synchronizing path generated reaches a final state. This semantical assumption can be ensured by structural properties of the automaton and/or the NMSPN. For instance, the time bounded Until of CSL guarantees this property. As a second example, the time unbounded Until of CSL also guarantees this property when applied on finite CTMCs where all terminal strongly connected components of the chain include a state that fulfills the target sub-formula of the Until operator. This (still open) issue is also addressed in [34, 51]. Due to this assumption, the random path variables are well defined and the expression associated to the specification may be evaluated with expectations defined w.r.t. the distribution of a random path conditioned by acceptance of the path. In other words, the LHA both calculates the relevant measures during the execution and selects the relevant executions for computing the expectations.

For a reader interested in statistical model checking, we mention several other tools dedicated to, or including a bit of, statistical model checking. This list is not exhaustive. The probabilistic model checker PRISM[42], originally only using numerical techniques now includes the possibility to check probabilities using statistical techniques. The well established timed model checker UppAal now has a statistical engine[22] in which timed systems are given a stochastic semantics and simulation is used to verify stochastic properties. The model checker Ymer[58] is fully dedicated to statistical model checking and can handle GSMPs as well as Markovian models, checking CSL properties. We can also mention VESTA[51], a statistical model checker that also computes expected values of several performance evaluation expressions, and PLASMA[36], also a statistical model checking platform that verifies (a variant of) bounded LTL properties.

3. Application of HASL to the analysis of stochastic oscillations

Oscillatory trends are a fundamental aspect of the dynamics of many biological mechanisms, therefore the ability to detect/measure them is crucial. In a stochastic modeling context, the aim is to be able to measure relevant aspects of oscillations such as the average amplitude/period of oscillations or, even better, the probability density function of oscillations' amplitude and period. In this section we discuss the application of the HASL method to measuring these aspects. As a first step, we look at the formal definition of what an oscillation is and provide two distinct characterizations upon which we base the construction of HASL specifications for assessing different aspects (i.e. period duration and

amplitude) of oscillatory traces. We then demonstrate the application of such HASL specifications on examples of stochastic oscillators in Section 3.2.

3.1. Measuring oscillations with HASL

Among various performance evaluation measures, the HASL formalism can be used for characterizing and effectively assessing oscillation related measures of DESP models. To this aim we consider two orthogonal approaches, i.e. two different automata-based characterizations of oscillatory traces. In the first one we introduce an LHA, $\mathcal{A}_{peaks}$, for assessing the peaks of so-called *noisy alternating* traces: i.e. traces exhibiting a (possibly infinite) sequence of local maxima interleaved with local-minima. The $\mathcal{A}_{peaks}$ automaton allows the user to assess different measures related to detected peaks such as, for example, the average value or the PDF/CDF of the peaks of a trace. In the second approach (based on [52, 53]), we introduce an LHA, denoted $\mathcal{A}_{per}$, for assessing the duration of so-called *noisy period* for *noisy-periodic* traces. i.e. traces that perpetually switch between states of a *low* to states of a *high* partition of the population domain of the observed species. The $\mathcal{A}_{per}$ automaton allows the user to assess different measures related to the periodicity of an oscillator such as, for example, the average duration as well as the *fluctuation* of the period of a stochastic oscillator. As we will see, both automata depend on certain configuration parameters and the chosen configuration establishes an *observational perspective* over the oscillating traces. For example with $\mathcal{A}_{per}$ the considered *high*/*low* partition of the observed species state space affects the measured period. In this respect we suggest a two-step approach should be followed: given an oscillating model, first the $\mathcal{A}_{peaks}$ LHA is applied, resulting in estimates of the peaks (hence the amplitude) of oscillations. Peaks measurements can then be used to identify a sensible partition of the domain of the oscillating species, hence to properly set up the parameters of the LHA for measuring *noisy periodic* traces. Figure 13 illustrates for example how, using automaton $\mathcal{A}_{peaks}$, it is possible to set a H threshold capturing 95% of the local maxima. An automaton using these thresholds will then be used to assess the duration of oscillation periods. We give details of both methods in the following.

Notation and vocabulary. Referring to the characterization of the traces of a DESP model, we introduce few basic notations and concepts we will refer throughout the remainder of this section. We denote σ an (infinitely long) simulation trace of an n-dimensional DESP model whose states' form is $s = (s_1, \dots s_n) \in \mathbb{N}^n$, with s_i being the value along the ith dimension (e.g the number of molecules of species i). We denote $\sigma_i(t)$ the i^{th} projection of $\sigma(t)$, the state in which σ is at time t. Following the characterization given in [52], we say that σ_i is either: *convergent* (i.e. tending to a finite value), *divergent* (i.e. tending to infinity) or *oscillating* (i.e. the lack of the previous two). Furthermore, σ_i is *periodic* with period $\delta > 0$ iff $\forall t, \sigma_i(t) = \sigma_i(t + \delta)$. Whenever there exists a positive minimal value with this property, it is called *basic period*. Thus σ is periodic oscillatory along the i-th dimension iff σ_i is both oscillating and periodic. We point out that, for stochastic models, "exact" periodicity (as given

above) is too restrictive hence in the following (Section 3.1.2) we introduce the less restrictive definition of *noisy periodicity* [52].

3.1.1. Measuring noisy alternating traces

We consider the problem of detecting the local maxima (minima) on traces of a given DESP model. Since traces of a DESP consist of discrete increments/decrements of at least one unit, it is up to the observer to establish what should be accounted for as a local maximum (minimum) during such detection process. In the most general case every single point corresponding to a change of trend is accounted as a maximum/minimum, even if distanced of only one unit from the previous one. In this case, however, the detected maxima (minima) will include the minimal peaks corresponding to stochastic noise, hence actually biasing local maxima (minima) related measures with the effect of stochastic noise. As a consequence, we propose a *parametric* characterization (Def. 3.1) of local maxima (minima) based on a so-called *noise* parameter which establishes an *observational perspective*: a point corresponding to a change of trend is detected as local maximum (minimum) only if its distance from its predecessor is at least equal to the value of the *noise* parameter. We then introduce the automaton $\mathcal{A}_{peaks}$ which encodes such a characterization and provides us with an effective machinery to detect local maxima/minima over traces of a DESP model.

Definition 3.1. *Let σ be a trace of an n-dimensional ($n \geq 1$), possibly infinite-state, DESP, and let $\delta \in \mathbb{N}_{\geq 0}$ represent a noise level. Trace σ_i is said* noisy periodically alternating *(or noisy alternating for short), with respect to noise level δ, iff it perpetually alternates from a local minimum to a local maximum and the minimal distance between consecutive minima and maxima is no smaller than δ.*

The detection of local maxima/minima along an alternating trace is illustrated in Figure 4.

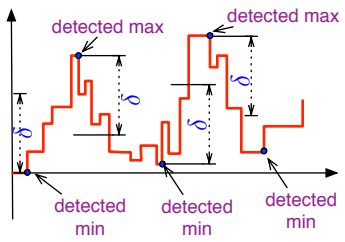


Figure 4: Example of *noisy alternating* trace corresponding to a given noise level (δ).

The $\mathcal{A}_{peaks}$ automaton: We introduce an LHA (Figure 5), denoted $\mathcal{A}_{peaks}$, designed for detecting local maxima/minima along noisy alternating traces of

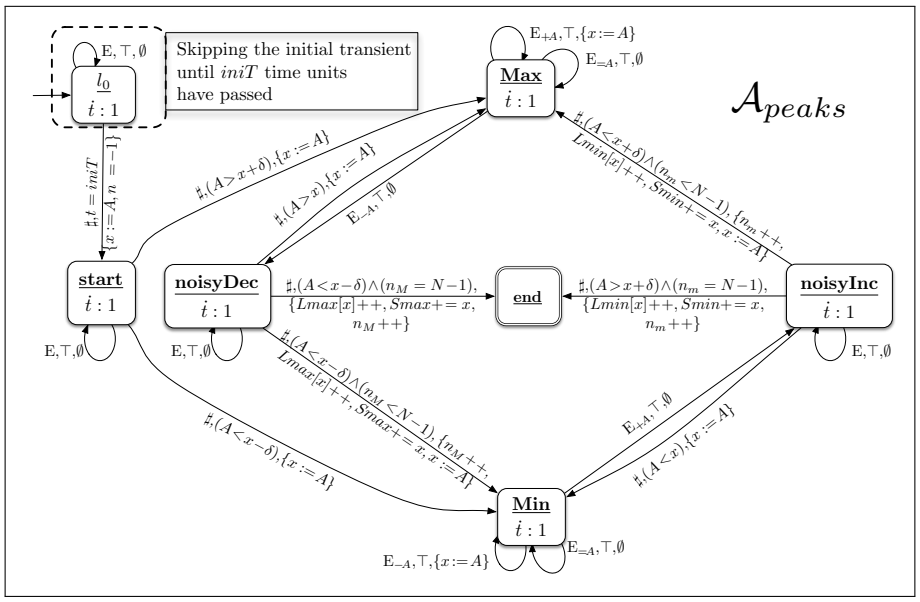


Figure 5: $\mathcal{A}_{peaks}$: an LHA for detecting local maxima/minima (for observed species A) of noisy periodic traces with level of noise δ.

a given observed species (in this case A). The automaton $\mathcal{A}_{peaks}$ is parametric with respect to a parameter δ. This parameter represents the level of noise tolerated in the system: after a local maximum, if the population decreases of less than δ, we keep looking for a higher local maximum. If the decrease is greater than δ then the maximum has been found and the automaton now looks for the next local minimum. Given the observed quantity A, the definition of $\mathcal{A}_{peaks}$ requires partitioning the set of events in three subsets: $E = E_{+A} \cup E_{-A} \cup E_{=A}$, where E_{+A} (respectively $E_{-A}, E_{=A}$) is the set of events resulting in an increase (respectively decrease, no effect) of the population of A.

The rationale behind the structure of $\mathcal{A}_{peaks}$ is to mimic the cyclic structure of an alternating trace through a loop of four locations, two of which (i.e. **Max** and **Min**) are targeted to the detection of local maxima, resp. minima. The simulated trace yields the automaton to loop between **Max** and **Min**, registering the minima/maxima along the path. The detailed behavior of $\mathcal{A}_{peaks}$ is as follows. Processing of a trace starts with a configurable filter of the initial transient (represented as a box in Figure 5). During the first $initT$ time units, the simulated trace simply grows without any peak detection[2]. The actual analysis begins in location **start** from which we move to either **Max** or **Min** depending whether we initially observe an increase (i.e. $x < A-\delta$) or a

[2]This is useful for eliminating the effect of the *initial transient* from long run measures as already discussed in [4].

Data variables			
name	domain	update definition	description
t	$\mathbb{R}_{\geq 0}$	*reset*	time elapsed since beginning measure (first non-spurious period)
$n_M(n_m)$	$\mathbb{N}$	*increment*	counter of detected local maxima (n_M), minima (n_m)
x	$\mathbb{N}$	*current value of* observed species A	(overloaded) variable storing most recent detected maximum/minimum
$Smax(Smin)$	$\mathbb{N}$		sum of detected maxima (minima)
$Lmax[](Lmin[])$	$\mathbb{N}^n$		array of frequency of heights of detected maxima (minima)

Table 1: The data variables of automaton $\mathcal{A}_{peaks}$ of Figure 5 for locating the peaks of a noisy oscillatory traces

decrease (i.e. $x > A+\delta$) of the population of the observed species A beyond the chosen level of noise δ. Once within the **Max**→**noisyDec**→**Min**→ **noisyInc** loop the detection of maxima/minima begins. Location **Max** (**Min**) is entered from **noisyInc** (**noisyDec**) each time a sufficiently large (w.r.t. δ) increment (decrement) of A is observed. On entering **Max** (**Min**), we are sure that the current value of A has moved up (down) of at least δ from the last value stored in x while in **Min** (**Max**), hence that value (x) is an actual local minimum (maximum) thus we add it up to $Smin$ ($Smax$). Then we increment the frequency counter corresponding to the level of the detected minimum $Lmin[x]$ (maximum $Lmax[x]$)[3] before storing the new value of A in x and finally increasing n_M (n_m), the counter of detected maxima (minima). Once in **Max** (**Min**) we stay there as long as we observe the occurrence of reactions which do not decrease (increase) the value of A, hence either a reaction of E_{+A} (E_{-A}), in which case we also store the new increased (decreased) value of A, hence a potential next local maximum (minimum) in x, or one of $E_{=A}$. On the other hand, on occurrence of a "decreasing" ("increasing") reaction E_{-A} (E_{+A}) we

[3]with a slight abuse of notation we refer to $Lmin[]$ and $Lmax[]$ as arrays whereas in reality within COSMOS/HASL they correspond to a set of variables $Lmin_i$, $Lmax_j$, each of which is associated to a given level of the observed population, thus $Lmin_1$ counts the frequency of observed minimum at value 1, $Lmin_2$ the observed minima at value 2 and so on. The number of required $Lmin_i$, $Lmax_j$ variables, which is potentially infinite, can be actually bounded with a negligible loss of precision to a sufficiently large value $Lmin_m$ (*resp.* $Lmax_m$) which must be established manually beforehand, for example by observing few previously generated traces.

move to **noisyDec** (**noisyInc**) from which we can either move back to **Max** (**Min**), if we observe a new increase (decrease) that makes the population of A exceed x (x exceed A), or eventually enter **Min** (**Max**) as soon as the observed decrease (increase) goes beyond the chosen δ (see above). For the automaton $\mathcal{A}_{peaks}$ depicted in Figure 5, the analysis of the simulated trace ends, by entering the **end** location either from **noisyDec** or **noisyInc**, as soon as N maxima (or minima, depending on whether the first observed peak was a maximum or a minimum) have been detected. Notice that $\mathcal{A}_{peaks}$ can straightforwardly be adapted to different ending conditions.

Measures associated with automaton $\mathcal{A}_{peaks}$. We introduce the following four target measures, given in terms of HASL expressions, associated with $\mathcal{A}_{peaks}$. All of them are computed on paths that end as soon as at least N minima or N maxima have been detected, so the measures only concerns transient behaviors up to seeing N extrema of the same type. Observe that a time-bounded version of $\mathcal{A}_{peaks}$ can be straightforwardly obtained in order to ensure termination (see discussion in Section 3.1.3).

- $Z_{max} \equiv E[last(Smax)/n_M]$: corresponding to the expected value of the average height along a path of the maximal peaks.
- $Z_{min} \equiv E[last(Smin)/n_m]$: same as Z_{max} but for minima.
- $Z_{PDFmax} \equiv PDF(last(Smax)/n_M, s, l, h)$: corresponding to the PDF of the average height (along a path) of the maximal peaks
- $Z_{PDFmin} \equiv PDF(last(Smin)/n_m, s, l, h)$: same as Z_{PDFmax} but for minima
- $Z_{Lmax} \equiv E(last(Lmax[k])/n_M)$: expected proportion of maximal peaks of height k along a path. This measure is used to compute the PDF of the height (along a path) of the maximal peaks
- $Z_{Lmin} \equiv E(last(Lmin[k])/n_m)$: same as Z_{Lmax} but for minima.

3.1.2. Measuring noisy periodic traces

Here we consider the problem of studying the periodicity of traces of a DESP model. As mentioned earlier, exact periodicity never occurs in practice in non degenerative stochastic systems. Following the approach of [53], we introduce a less restrictive characterization of periodicity, namely the so-called *noisy periodicity* (see Definition 3.2) which is based on two thresholds parameters, L and H. As a consequence, the perceived noisy periods depends on the chosen L and H thresholds.

Definition 3.2. *Let σ be a trace of an n-dimensional ($n \geq 1$), possibly infinite-state, DESP, and let $L, H \in \mathbb{N}$ ($0 \leq L < H$) be a lower, respectively an upper bound, inducing intervals $low = (-\infty, L]$, $mid = (L, H)$ and $high = [H, +\infty)$, trace σ_i is said* noisy periodic *with minimal amplitude $H - L$ iff it perpetually switches from low to high (possibly passing through mid) and returning to low.*

Remark 3.1. *A trace that is noisy periodic is also necessarily noisy periodically alternating (i.e. noisy-periodicity is a sufficient condition for noisy-periodic-alternance), however the opposite is not the case: a periodically alternating trace may well diverge.*

According to Definition 3.2 a noisy-periodic trace corresponds to the following ω-regular expression[4]:

$$e_{np} = (low(low + mid)^* high(high + mid)^*)^\omega \tag{1}$$

Notice that such a characterization induces a bound on the minimal-amplitude of oscillations (corresponding to the width of the chosen partition, i.e. the distance H–L), but entails no specific constraints on the maximal amplitude nor on the period duration of oscillations (i.e. traces belonging to this class will include also those with non-constant period duration, i.e. those with noisy-period). It is also worth mentioning that, since we use an approach based on sampling, we will not be able to study sustained (i.e. perpetual) oscillations and we thus develop different ways to study oscillating characteristics on bounded paths.

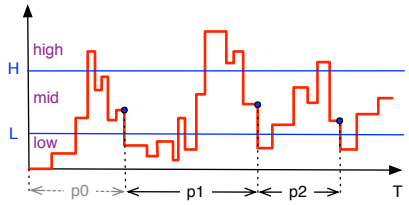


Figure 6: Example of *noisy periodic* trace corresponding to a given ($L < H$) partition of a DESP state space.

Figure 6 shows an example of a noisy periodic trace of a DESP. A period corresponds to the behavior of the system between two consecutive crossing points corresponding to entering in one extreme of the partition, interleaved by at least one crossing point into the opposite extreme. For example in Figure 6 the first period p_1 corresponds to the duration between the first *mid*-to-*low* crossing and the successive *mid*-to-*low* crossing interleaved by a *mid*-to-*high* crossing. Notice that the first complete period(p_1) might be preceded by a spurious period (i.e. p_0) which should be discarded as there's no guarantee that $t = 0$ corresponds with an actual *mid*-to-*low* crossing, hence with the actual beginning of p_0.

The $\mathcal{A}_{per}$ automaton: We introduce an LHA (Figure 7), denoted $\mathcal{A}_{per}$, designed for detecting and measuring the periods of a given observed species (in this case A).

[4]assuming the oscillator starts in *low*: slight variants of (1) hold in case the oscillator starts in either *mid* or *high*.

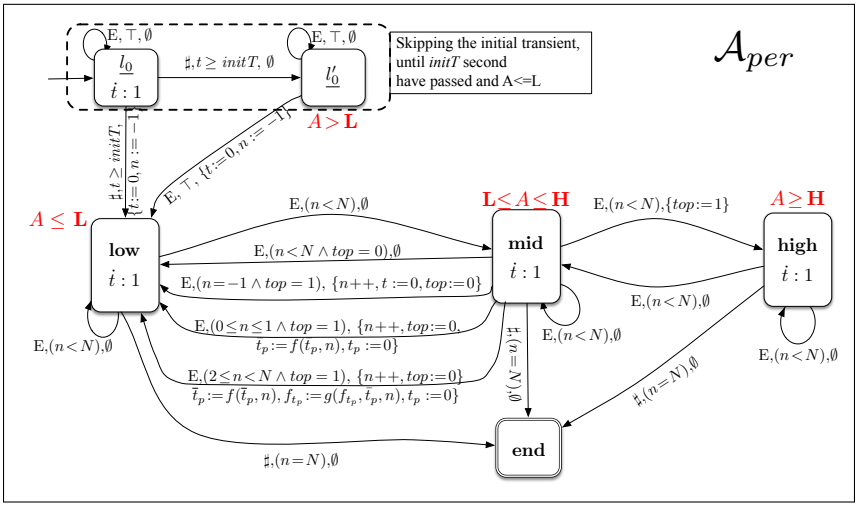


Figure 7: $\mathcal{A}_{per}$: an LHA for selecting noisy periodic traces (with respect to an observed species A) related to partition $low = (-\infty, L]$, $mid = (L, H)$ and $high = [H, +\infty)$.

More specifically, $\mathcal{A}_{per}$ is designed for assessing two characteristics of an oscillatory trace: the mean period duration (denoted $\bar{t}_p$) and the period fluctuation (denoted f_{t_p}) over the first N periods detected along a trace. Period fluctuation represents how much (on average) a single period duration deviates from the mean duration computed over the N periods observed along a trace. From the point of view of analysis, the period fluctuation is a useful indication of the irregularity of the observed oscillator. Automaton $\mathcal{A}_{per}$ consists of an initial *transient filter* (locations l_0, l'_0) plus three main locations **low**, **mid** and **high** (corresponding to the partition of A's domain induced by thresholds $L < H$). As with $\mathcal{A}_{peaks}$, the simulated trajectory initially unfolds for a given duration ($initT$), letting $\mathcal{A}_{per}$ within l_0 without doing anything. After $initT$ time units, $\mathcal{A}_{per}$ enters **low**[5] (possibly through l'_0 if at that time the population is above level L) where the actual oscillation analysis begins. From **low** the automaton follows the profile of A hence moving to **mid** as soon as $L < A < H$ holds, and then to **high** as soon as $A \geq H$. With $\mathcal{A}_{per}$, a period starting point is associated with the first *mid*-to-*low* crossing that follows a *mid*-to-*high* crossing[6]. Hence the first detected period (crossing from **low** to **high** and back to **low**) is discarded as its duration may be spurious. $\mathcal{A}_{per}$ uses six variables (Table 2): t is a timer that keeps track of simulation time; n counts the number of detected noisy-periods while *top* is a boolean flag used for distinguishing the **mid** to **low** crossing points that correspond to the closure of one period (i.e. when a

[5]the choice of beginning measuring in **low** is arbitrary, equivalently $\mathcal{A}_{per}$ can be defined so that analysis starts in either **mid** or **high**.

[6]again equivalent versions of $\mathcal{A}_{per}$ can be easily obtained which detect periods by considering different starting points, e.g periods that start with a crossing from *mid* to *high*.

name	domain	update definition
t	$\mathbb{R}_{\geq 0}$	*reset*
n	$\mathbb{N}$	*increment*
top	bool	*complement*
t_p	$\mathbb{R}_{\geq 0}$	*reset*
$\bar{t}_p$	$\mathbb{R}_{\geq 0}$	$f(\bar{t}_p, t_p, n) = \frac{1}{n+1}(\bar{t}_{p_n} \cdot n + t_p)$
f_{t_p}	$\mathbb{R}_{\geq 0}$	$g(f_{t_p}, \bar{t}_p, t_p, n) = \frac{1}{n}[(n-1)f_{t_p} + (t_p - \bar{t}_p)(t_p - f(\bar{t}_p, t_p, n+1))]$

Table 2: The data variables of automata $\mathcal{A}_{per}$ of Figure 7 for measures of noisy-periodicity

traversal from *mid*-to-*low* has been preceded by a traversal from *high*-to-*mid*) from those which do not. Notice that, in order to ignore the first potentially spurious period (p_0 in Figure 6), n is initially set to -1 on entering **low** from the initial transient filter, and the simulation time t is then reset on detection of the closure of the first spurious detected (i.e. on entering **low** from **mid** when $n = -1$). Furthermore t_p stores the duration of the last detected period, while $\overline{t_p}$ maintains the mean duration of all (so far) detected periods and f_{t_p} stores the fluctuation of the period duration for all (already) detected periods. Notice that the fluctuation (i.e. f_{t_p}) is computed *on the fly* (see Table 2) by adaptation of the so-called *online algorithm* [40] for computing the variance out of a sample of observations. Finally the analysis of simulated trajectories stops (by entering the accepting location **end**) as soon as the N^{th} period has been detected.

Measures associated with automaton $\mathcal{A}_{per}$. We introduce the following four target measures, given in terms of HASL expressions, associated with $\mathcal{A}_{per}$:

- $Z_1 \equiv E[last(\bar{t}_p)]$: corresponding to the mean value of the period duration for the first N detected periods.
- $Z_{2_a} \equiv PDF(\bar{t}_p), s, l, h)$: corresponding to the PDF of the average period duration over the first N detected periods, where $[l, h]$ represents the considered support of the estimated PDF, and $[l, h]$ is discretized into uniform subintervals of width s
- $Z_{2_b} \equiv CDF(\bar{t}_p), s, l, h)$: as Z_{2_a} but corresponds to the CDF of the average period duration over the first N detected periods.
- $Z_3 \equiv E[last(f_{t_p})]$: corresponding to the fluctuation of the period duration.

Expressions Z_1, Z_{2_a} and Z_{2_b} represent measures related to the duration of the oscillation periods (i.e. the mean value, and the PDF, respectively the CDF, of the period duration). On the other hand Z_3 is designed to asses the *fluctuation* of the period duration, i.e. how much the N periods detected along a trace differ from their average duration. Observe that the measured period fluctuation (i.e. Z_3) provides us with a useful measure of the *irregularity*, from

the point of view of the period duration, of the observed oscillation. Thus an oscillator whose trajectories are likely to exhibit periods of, more or less, the same duration (e.g. Figure 10(a)) will result in smaller fluctuation f_{t_p} than an oscillator whose trajectories exhibits periods of more variable duration (e.g. Figure 10(b)). We give an example of analysis of the regularity of the oscillation periods on a model of the repressilator (see Section 3.2).

Remark. As we argued before, the measurements of the period of oscillations obtained through automaton $\mathcal{A}_{per}$ are affected by the considered L, H partition, i.e. the chosen value of the L, H parameters of $\mathcal{A}_{per}$. We provide an example illustrating this aspect in Section 3.2.2 where we analyze the repressilator model. The problem is that, without any prior knowledge about the observed oscillator, and in particular about the height of the maximal/minimal peaks of the corresponding traces, we risk to set the L, H parameters to values which are not representative of the actual observed oscillator. For example if we set H to a value which is well above the average height of the maximal peaks then we are going to miss most of the peaks. As a consequence we envisage that, when assessing oscillators, a protocol involving the $\mathcal{A}_{peaks}$ and $\mathcal{A}_{per}$ automata should be followed. More specifically: first apply $\mathcal{A}_{peaks}$ to get a measurement of the distribution of maximal/minimal peaks' values, then use this information to set reasonable values for the L, H parameters of $\mathcal{A}_{per}$. We provide details about the application of such a protocol through an example presented in Section 3.2.1.

3.1.3. Discussion

With respect to the HASL based oscillation analysis approach introduced above, there are a number of delicate aspects which are worth discussing.

Measuring vs verifying. First it is worth stressing that the HASL approach is not a "pure" *verification* approach, in the sense of classical model checking. A "pure" *verification* approach would allow to decide whether a given stochastic model oscillates or not. In stochastic modeling deciding about a *qualitative property* (such as *the model oscillates*) boils down to comparing a *quantitative measure* (of probability) against a given threshold. For oscillations this has been shown to be feasible in some cases, for example: whether a finite-state CTMC model oscillates sustainably can be verified [53] but requires the computation of the steady-state distribution of (a timed-automaton expanded version of) the considered CTMC[7] (thus is affected by the state-space explosion problem). HASL, as any other statistical model checking approach, is inherently limited to reasoning about finite observations, hence is not suited for *verification* of *infinite behaviors*, such as sustained oscillations. Nonetheless, it can be used to extract significant (finite horizon) knowledge which can be used as an indicator of a model's *infinite behavior*. Thus, if a sustained oscillator is analyzed through

[7] in fact it requires verifying that the steady-state measure for certains states of the timed-automaton expanded CTMC add up to 1.

HASL, its oscillations will be captured by the $\mathcal{A}_{per}$ and $\mathcal{A}_{peaks}$ automata whose parameters will be appropriately set so as to discard a possible non-oscillatory initial transient. On the other hand, if a non-oscillating model is studied using the $\mathcal{A}_{per}$ and $\mathcal{A}_{peaks}$ automata, the number of detected periods will either be null (if the model does not include any period in its initial transient), or it will reach an asymptotic value as we enlarge the observation windows (if the model include some initial periods before either converging or diverging).

On automata's parameters and termination. With respect to termination, observe that $\mathcal{A}_{per}$ and $\mathcal{A}_{peaks}$ have been, for the sake of simplicity, presented (Figure 7 and Figure 5) as *event-bounded* measures, thus measuring terminates once a finite number N of oscillation-related events have been observed (i.e. N noisy-periods for $\mathcal{A}_{per}$, N local maxima for $\mathcal{A}_{peaks}$). This may lead to non-termination (or to a very long runtime) when applied to certain models. To avoid non-termination issues one can simply consider the *time-bounded* version of both automata, denoted $\mathcal{A}_{per}^{t\leq T}$ and $\mathcal{A}_{peaks}^{t\leq T}$, obtained by adding a time constraint $t \leq T$ (T being a parameter of the automaton) to each arc of the automaton (hence enforcing the synchronization to terminate, at latest, after T time units).

Another important aspect is the setting of the automata's parameters (L and H for $\mathcal{A}_{per}$, δ for $\mathcal{A}_{peaks}$) as these establish an *observational perspective*, hence affect the outcome of the measure. For $\mathcal{A}_{per}$, the further H (resp. L) is chosen above (resp. below) the actual average value of maximal (resp. minimal) peaks of the considered oscillator, the less likely it becomes for a trace to enter the $[H, \infty)$ (resp. $[0, L)$) region. A simple way to chose "reasonable" L and H values is to plot a few single traces generated for the considered model and to fix the values of L and H close to the peaks as perceived on the traces plot. A more accurate manner to proceed is to first employ $\mathcal{A}_{peaks}$ for measuring the PDF of the heights of peaks and then exploit such measure to set $\mathcal{A}_{per}$'s L and H values. Since the outcome of an $\mathcal{A}_{peaks}$ measure depends on the chosen δ parameter (i.e. the height of noisy spikes to be filtered out from the detection of local maxima/minima), the measurement of the PDF of peaks should be repeated, starting form $\delta = 1$ and for increasing values of δ. The comparison of results obtained for different small values of δ, (e.g. $\delta = 1$ to $\delta = 5$) will provide information about the localization of noisy spikes, thus allowing for figuring out what is the most likely height of the "actual" peaks of oscillation. In Section 3.2 we illustrate the combined application of $\mathcal{A}_{peaks}$ and $\mathcal{A}_{per}$ measurements on a concrete example.

Initial transient. Another relevant aspect is related to the fact that what we aim at is measuring relevant characteristics of oscillations (e.g. the average duration of the oscillation period) but only once the system has reached its long term behavior, hence disregarding the (spurious) effect that the *initial transient* might have on the measured outcome. The elimination of the effect of the initial transient from measures obtained via simulation of a stochastic model is a well established subject in the literature [26]. In this respect it would be

desirable to equip the COSMOS tool with facilities for automatically detecting the duration of the initial transient and for the elimination of its effects from the target measure. In this paper we only provide a partial (not automatic) scheme to tackle that issue. In particular we equip each LHA dedicated to assessing oscillations with an *initial transient filter*, i.e. an elementary part of the automaton whose only goal is to let time pass (while simulating a trace) before actually starting the measuring process. Thus the user should first run a number of preliminary experiments (i.e. reiterated with increasing values of the $initT$ parameter) aimed at establishing at which point the estimate of the target measure starts stabilizing . For example our experiments indicated that in the case of the repressilator model of Section 3.2 the initial transient has no effect on the measures of the period as such measures turn out to be independent of $initT$. A similar question arises with the appropriate length of paths in order to have a measure that is close to what could be computed on infinite paths. Figure 17 shows how, by increasing the length of observed path, we see a stabilization in the measure that gives us a good idea of the long term behavior.

3.2. Case study: measuring oscillations of a three-genes repressilator model

To demonstrate the above discussed approach, we have considered different examples of oscillators. As a first step towards the empirical validation of HASL based measurement of oscillations, we have considered a toy model, i.e. a probabilistic square wave oscillator. The goal in that respect was to have a simple model for which an analytical expression, for both the mean value and the fluctuation of the cycle duration, existed. Hence we have compared measurements obtained through $\mathcal{A}_{per}$ with the corresponding exact values computed analytically for the square wave oscillator: the estimates obtained with HASL turned out to precisely match the exact values (for details see Appendix A.1). In the remainder of this section, on the other hand, we report on the application of HASL based oscillation analysis to a realistic model of genetic oscillator, known as the repressilator [27].

3.2.1. A three-genes repressilator

Repressilator model. We consider a model of the three-genes repressilator, a simple genetic network with three elements inhibiting the production of each other in a cyclic mono-directional manner. The repressilator is a real genetic circuit, consisting of the natural repressor proteins and the genes coding for them. The cyclic topology gives a certain robustness to the circuit (Fig. 8) that exhibits oscillations for a broad range of the parameters' values.

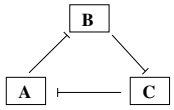


Figure 8: The cyclic topology of the repressilator circuit.

Here we consider a version of the repressilator model taken from [47] which essentially is a simplified version of the more general model of Elowtitz [27]. In our model, gene expression is considered as a single step process occurring when the RNA-polymerase ($RNAp$) reacts with the promoter (Pro_i). The result of this reaction is the appearance of the newly synthesized protein molecule (P_i) and the release of the reactants, *i.e.* promoter and RNA-polymerase. The i-th promoter (where $i \in \{1, 2, 3\}$) yields the corresponding protein (P_i), and this is modeled by reaction (2).

$$Pro_i + RNAp \xrightarrow{k} RNAp + Pro_i + P_i \tag{2}$$

Notice that in our model $RNAp$ is never consumed hence its only role is to control the speed of the expression of the three genes (higher $RNAp$ population induces faster gene expression). The repression reactions are modeled as binding/unbinding of the repressor protein to/from the corresponding promoter providing the repressilator's cyclic topology. These reactions are shown in (3). Protein (P_j) binds to the promoter (Pro_i), forming the complex ($P_j Pro_i$) which prevent the further expression reaction for promoter Pro_i. The unbinding of the protein makes the promoter available for the expression again. In reactions (3), $i \in \{1, 2, 3\}$ and $j = i - 1$, except for $i = 1$, in which case $j = 3$.

$$\begin{aligned} P_j + Pro_i &\xrightarrow{k_r} P_j Pro_i \\ P_j Pro_i &\xrightarrow{k_u} P_j + Pro_i \end{aligned} \tag{3}$$

Finally, proteins undergo degradation (reactions (4)), which is modeled as a single step uni-molecular reaction. Protein degradation takes place either when proteins are free (with kinetic rate d), or when they are bound to promoters (with kinetic rate d'). The reactions (4) take the same values for i and j as in reactions (3).

$$\begin{aligned} P_j &\xrightarrow{d} \emptyset \\ P_j Pro_i &\xrightarrow{d'} Pro_i \end{aligned} \tag{4}$$

Within the stochastic interpretation, the reactions (2), (3) and (4) are associated to (negative) exponential distributions, hence the repressilator can be simulated following Gillespie's [30] approach. In the remainder we will mostly consider the following default configuration of the model parameters: $k = 0.01$, $k_r = 1$, $k_u = 0.08$, $d = d' = 0.01$, with $RNAp = 30$ being the initial population of RNA-polymerases and also assuming that the system contains only one copy of each gene, i.e. every promoter is present in a single copy. However we will also analyze the model's dynamics under perturbation of the following parameters: the $RNAp$ initial population, and the degradation rate (d') of proteins when attached to the promoter regions. The complete set of 15 reactions for the three-genes repressilator model is given by equations (5), while its encoding in GSPN form, which is what we used for our experiments with the Cosmos tool, is illustrated in Figure 9.

$$
\begin{aligned}
&R_{1_1} : Pro_1 + RNAp \xrightarrow{k} RNAp + Pro_1 + P_1 && R_{2_1} : P_1 + Pro_2 \xrightarrow{k_r} P_1Pro_2 \\
&R_{1_2} : Pro_2 + RNAp \xrightarrow{k} RNAp + Pro_2 + P_2 && R_{2_2} : P_2 + Pro_3 \xrightarrow{k_r} P_2Pro_3 \\
&R_{1_3} : Pro_3 + RNAp \xrightarrow{k} RNAp + Pro_3 + P_3 && R_{2_3} : P_3 + Pro_1 \xrightarrow{k_r} P_3Pro_1 \\
&R_{3_1} : P_1Pro_2 \xrightarrow{k_u} P_1 + Pro_2 && R_{4_1} : P_1Pro_2 \xrightarrow{d'} Pro_2 \\
&R_{3_2} : P_2Pro_3 \xrightarrow{k_u} P_2 + Pro_3 && R_{4_2} : P_2Pro_3 \xrightarrow{d'} Pro_3 \\
&R_{3_3} : P_3Pro_1 \xrightarrow{k_u} P_3 + Pro_1 && R_{4_3} : P_3Pro_1 \xrightarrow{d'} Pro_1 \\
&R_{5_1} : P_1 \xrightarrow{d} \emptyset && R_{5_2} : P_2 \xrightarrow{d} \emptyset \\
&R_{5_3} : P_3 \xrightarrow{d} \emptyset
\end{aligned}
\tag{5}
$$

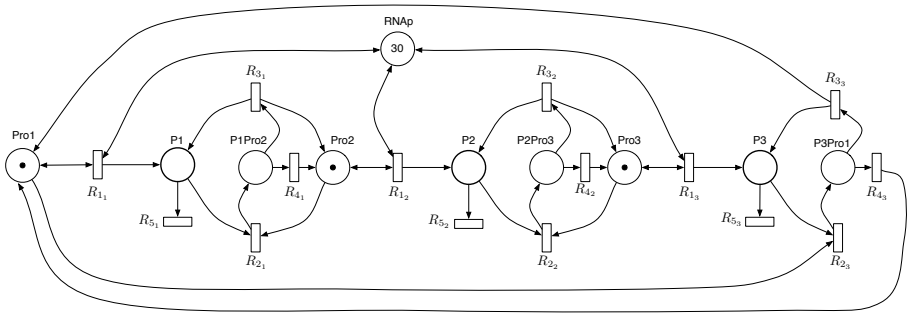
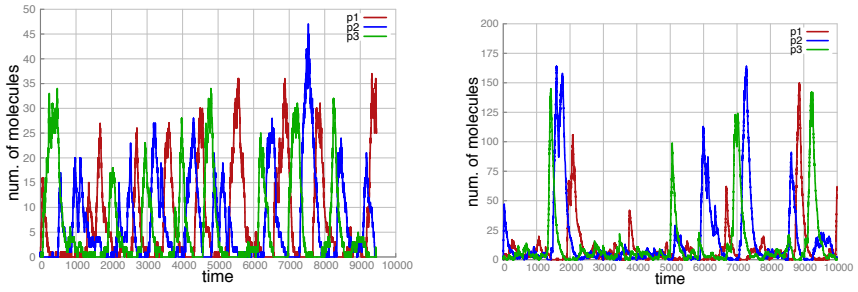


Figure 9: GSPN model of the three-genes repressilator: places P1, P2 and P3 represent the population of the three expressed proteins.

Figure 10 shows the dynamics of the three expressed proteins along a single simulation trace of the repressilator model referred to different initial populations of the *RNAp* (i.e $RNAp = 30$ and $RNAp = 200$). The three proteins' populations oscillate in a co-ordinated manner, however the curves are rather noisy (with respect to both amplitude and period duration). Observe that increasing the *RNAp* population, hence the speed of protein expression, induces a much more pronounced irregularity of the period duration (Figure 10(b)).

Strategies for the analysis of the repressilator model. As in most realistic stochastic oscillators, oscillation traces in the repressilator are very noisy (see Figure 10), and dependent on several parameters of the model (e.g. initial populations, decay rates, etc.). In Section 3.1.2 we have introduced the $\mathcal{A}_{per}$ automaton (Figure 7) as a means for measuring noisy periodic traces. However, the resulting measures are affected by the chosen L and H threshold parameters (i.e. intuitively: the larger the distance $H - L$ the longer the period duration). A simple manner to find out reasonable values for the L and H parameters consists in generating a simulation trace and manually establishing how to partition the



(a) initial $RNAp$ =30 (default configuration)

(b) initial $RNAp$ =200

Figure 10: A single simulation trace of the three-genes repressilator model with initial $RNAp$ =30 (left) and $RNAp$ =200 (right): observe that with higher initial $RNAp$ the trace consists of higher peaks grouped into rather irregularly distanced batches, hence, in general, higher $RNAp$ population seems to result in increased *irregular* oscillations.

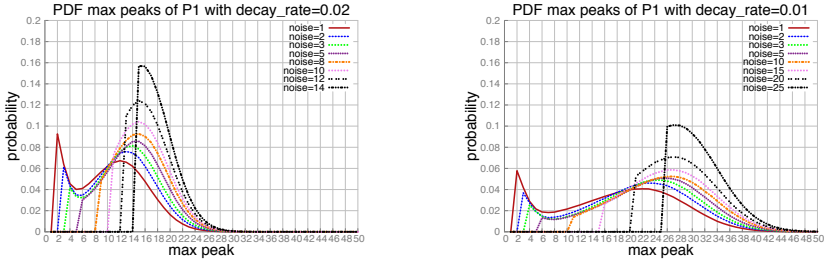
domain of the oscillating species so that the *high* (resp. *low*) interval contains the maximum (resp. minimum) peaks that we consider as those characterizing a period. Despite its simplicity, this approach has important drawbacks: first it is not formal (establishing the L and H values by observation of a single trace is a poorly accurate approach, with no control on the introduced approximation); second it is not automatized.

In the following we propose a methodology for establishing reasonable L and H values based on the measurement of the PDF of local maxima (minima) of the repressilator.

Detecting the most-likely local-maxima (minima) of the repressilator. Figure 11 and Figure 12 depict the PDF of the local maxima, respectively minima, of the repressilator measured through the $\mathcal{A}_{peaks}$ automaton (Figure 5). Such measures have been done for different values of the proteins' (i.e. P_1, P_2, P_3) decay rate: $d = 0.01$ (the "standard" decay) and $d = 0.02$ (a two times faster decay). To assess the effect that the chosen level of *noise* (i.e. the δ parameter of $\mathcal{A}_{peaks}$ has on the the outcome, we repeated the experiment for different values of $\mathcal{A}_{peaks}$'s *noise* parameter, hence both Figure 11 and Figure 12 contain several plots corresponding with different values of *noise*.

For the PDF of local maxima given in Figure 11, we can see that all plots exhibit a truncated bell-shape profile with truncation, as expected[8], corresponding to the chosen level of *noise*. Furthermore, for small values of *noise* the probability mass exhibits a truncation peak centred on the truncation point. Such a peak reflects the specific nature of the oscillations of the repressilator, and more

[8]peaks with height below the chosen *noise* are not detected by $\mathcal{A}_{peaks}$ hence their probability is correctly zero.



(a) PDF of local maxima with decay-rate 0.02.

(b) PDF of local maxima with decay-rate 0.01.

Figure 11: PDF of the local maxima of repressilator's oscillation peaks for different protein's decay rate and for different values of the accepted noise (i.e. δ) parameter of $\mathcal{A}_{peaks}$

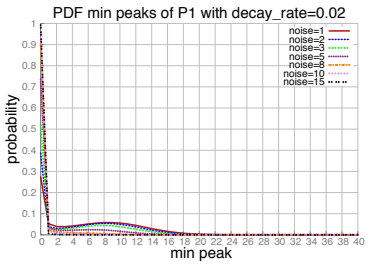
.

specifically it provides insights about where noisy small spikes are statistically located. By looking at single traces of the repressilator (e.g. Figure 10(a)), we observe that noisy small spikes appear to be more frequent around the minimal and maximal peaks of the oscillations than along the steep parts, thus when *noise* is set to small values such small spikes are also detected by $\mathcal{A}_{peaks}$ and this yields the spurious peak on the PDF.

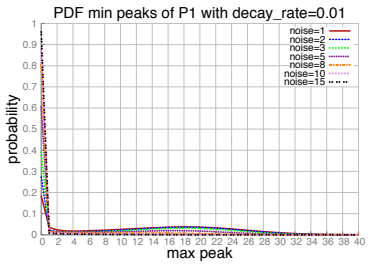
Observe further that for sufficiently large values of *noise* (i.e. $noise \geq 5$) the PDF plots are equally centered on a value (≈ 15, for $decay\_rate = 0.02$, and ≈ 26, for $decay\_rate = 0.01$) which corresponds with the average of the detected local maxima (see Figure 14(a)).

For the PDF of local minima shown in Figure 12, (almost) all plots exhibit a monotonically decreasing profile with a common maximum at zero, which indicates that the most likely minimum of each oscillation period is indeed located at zero (in line with the profile of single traces such as e.g. Figure 10). However, for low values of *noise*, the probability mass (Figure 12(d) and Figure 12(c)) actually includes a little bump slightly below the region where the local maxima probability mass is concentrated (≈ 15, for $decay\_rate = 0.02$, and ≈ 26, for $decay\_rate = 0.01$). Again such bump is a reflection of the stochastic character of the repressilator and indicates that little noisy spikes are also more densely located towards the upper peaks of oscillations. Observe that such bumps correctly fades away as we increase the level of *noise* in our experiments (i.e. we filter out the noisy spikes from the measure).

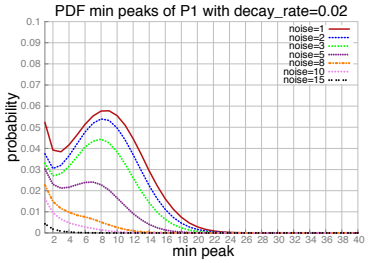
Exploiting the PDF of maxima/minima to set the L,H thresholds. An issue with the $\mathcal{A}_{per}$ automaton is that it requires to set the L,H threshold parameters, whose value has also an effect on the measured periods (see Figure 17(a) for example). Here we propose that appropriate values of L and H should be chosen as a function of the distribution of probability of the detected local



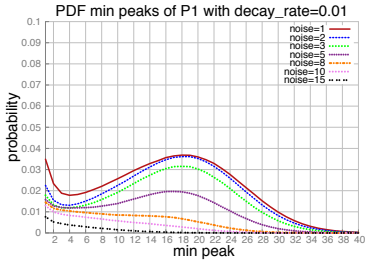
(a) PDF of local minima with decay-rate 0.02



(b) PDF of local maxima with decay-rate 0.01.



(c) details of PDF of local minima with decay-rate 0.02



(d) details of PDF of local maxima with decay-rate 0.01.

Figure 12: PDF of the local minima of repressilator's oscillation peaks as a function of the decay rate and for different values of the accepted noise.

maxima and minima estimated with the $\mathcal{A}_{peaks}$ automaton. Intuitively for a given probability value p (i.e. quantile) we chose H (L) so that the probability that local maxima (minima) are above (below) H (L) is at least p. This coincides with determining the quantile of the CDFs of local maxima (minima). Figure 13 shows the reverted CDFs (i.e. $(1 - CDF)$) of local maxima (straightforwardly derived from the PDFs in Figure 11). By choosing $p \in (0, 1)$ and looking at the x-coordinate of the intersection points of each curve with $y = p$ we obtain the desired value for H. For example, by setting $H = 13$ (x-coordinate of the intersection of yellow curve with $p = 0.95$) we know that with 95% chance, when protein P_1 decays with rate 0.01, the local maxima of oscillations of minimal amplitude 10 (i.e. $noise = 10$) will be above H.

3.2.2. Studying the effect of protein degradation on the repressilator oscillations

By application of automata $\mathcal{A}_{per}$ and $\mathcal{A}_{peaks}$, we have analyzed the effect that the decay rate (denoted d_1 hereinafter) of protein P_1 has on the repressilator oscillations. Speeding up d_1 corresponds to augmenting the throughput of P_1 degradation, thus intuitively should result in a lower P_1 average population, hence a lower height of maximal peaks of P_1 oscillation, and, as a consequence,

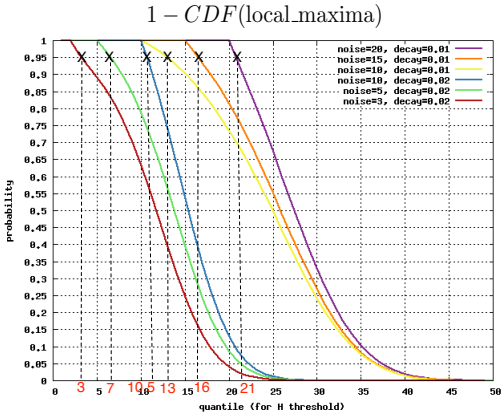
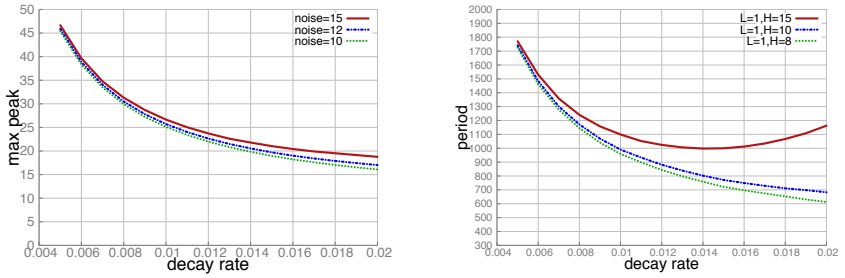


Figure 13: Setting the H threshold parameter with respect to the 95% quantiles of the CDF of the local maxima of the repressilator. On the x axis, in red, the values above which the local maxima of oscillations fall with 95% chance (e.g. with $decay\_rate = 0.01$ the 95% of the local maxima, which are at least $noise = 10$ units distant from the preceding local minimum, will be greater or equal to 13. Thus, by setting $H = 13$, we are guaranteed to miss out no more than 5% of all observed such local maxima.)

also a shorter period duration (i.e. the delay between two peaks should shorten as the height of peaks decreases). Conversely, slowing down d_1 should have the opposite effect (i.e. a higher average population, hence higher oscillation peaks, hence longer period duration). Figure 14 depicts the measures of the average height of maximal peaks (Figure 14(a)) and of the average period duration (Figure 14(b)) computed through $\mathcal{A}_{peaks}$, respectively $\mathcal{A}_{per}$ for $d_1 \in [0.005, 0.02]$. The measured height of maximal peaks (assessed for different values of the noise parameter of $\mathcal{A}_{peaks}$) monotonically decreases thus confirming the intuition that peaks shrink as P_1 degradation speeds up.

As for the period duration (Figure 14(b)), we can notice the effect that the *observational perspective*, i.e. the chosen L, H parameters of $\mathcal{A}_{per}$, has on the target measure, i.e. the average period duration. The average period measured for trajectories with maximal peaks above 8 and 10 (i.e. curves H=8 and H=10) monotonically decreases over $d_1 \in [0.005, 0.02]$ thus confirming the intuition that the period of oscillation shrinks as we speed up P_1's degradation. However, if we increase the threshold and only consider trajectories with peaks above $H = 15$, the measured period turns out not to be monotonically decreasing but rather displays a minimum at approximately $d_1 = 0.015$. This, which may look as an anomaly, is actually a sensible consequence of the configuration of the H parameter of $\mathcal{A}_{per}$. What happens in reality to the P_1 signal as we increase d_1 is that the peaks of oscillations are loosing height hence they are less likely to go above a $H = 15$. On the hand with $H = 15$ the period is computed only based on these (less likely) high peaks. This artificially increases the computed period.



(a) average height of maximal peaks.

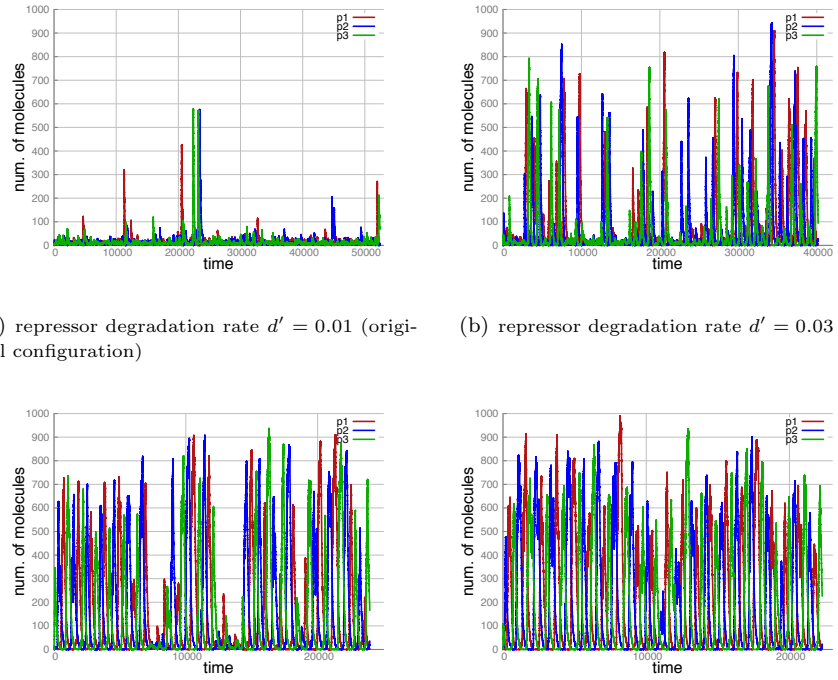
(b) average duration of a period.

Figure 14: Average height of maximal peaks (left) and period duration (right) measured with $\mathcal{A}_{peaks}$, respectively, $\mathcal{A}_{per}$ on the repressilator with initial $RNAp=30$ and in function of the degradation rate of protein P_1

3.2.3. Assessing the robustness of the repressilator oscillations

As an application of the measurement of the fluctuation of the cycle duration using HASL, we considered the problem of analyzing how robust the oscillations of the repressilator are with respect to certain parameters of the model, in particular: 1) the rate at which proteins are expressed (which in our model is proportional to the population of RNAp) and 2) the rate d' at which proteins degrade while attached to the promoter of the gene they repress (i.e. the rate of the reactions $Pro_jP_i \xrightarrow{d'} Pro_j$). We have already pointed out (Figure 10) that an increase of the protein expression speed (i.e. through RNAp population) induces an increased irregularity of the periodicity of the peaks. Such an irregularity is explained as follows: with faster expression proteins have a tendency to rapidly accumulate (in places P_1, P_2, P_3), hence increasing the repression of the controlled protein by constantly occupying the corresponding promoter (higher throughput of reactions R_{2_x}). One way for compensating the effect of faster expression is simply to speed up protein degradation, for example through rate d'. Figure 15 shows traces for configurations corresponding to a factor 33 increase of the expression speed (i.e. $RNAp = 1000$) and for different values of degradation rate d'. In absence of compensation ($d' = 0.01$ as in default configuration) a 33x speedup of the expression process results into a degenerated signal consisting of irregularly scattered spikes, superposed to a noisy base (Figure 15(a)). However, as we increase d' we progressively re-establish the oscillatory regime of the repressilator (Figure 15(d)).

If the traces in Figure 15 illustrate experimentally the (ir)regularity of oscillations, it is possible to measure this irregularity using HASL, as we will explain later. Figure 16 shows the value of the *average cycle duration* and the *average cycle fluctuation*, measured (through automaton $\mathcal{A}_{per}$, expressions Z_1, resp. Z_3, and using $L=1, H=100$ as thresholds for detecting periods) for the speeded up (i.e. $RNAp$ =1000) model and for different values of the degrada-



(a) repressor degradation rate $d' = 0.01$ (original configuration)

(b) repressor degradation rate $d' = 0.03$

(c) repressor degradation rate $d' = 0.05$

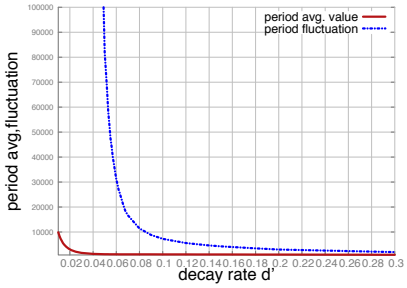
(d) repressor degradation rate $d' = 0.08$

Figure 15: traces of the repressilator corresponding to a 33x faster protein expression ($RNAp$ =1000) and for different values of protein's degradation rate d' (i.e the speed of the degradation of the protein P_i when attached to the promoter).

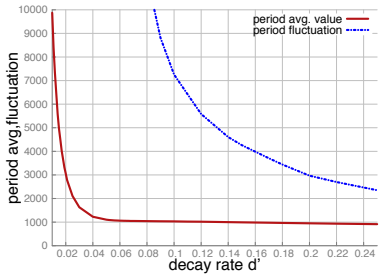
tion rate $d' \in [0.01, 0.3]$. From plots in Figure 16 we observe that both the average value and fluctuation of the period monotonically decrease over the interval $d' \in [0.01, 0.3]$. The compensating effect that the degradation rate d' has over the irregularity of the oscillations is highlighted by comparison of the two curves in Figure 16(b). In fact for $d' \in [\sim 0.6, \sim 0.25]$ the period duration remains roughly constant ($\sim$1000) whereas the period fluctuation exhibits a substantial decrease. Such a fluctuation decrease corresponds to the regularization of oscillations, illustrated by the trajectories in Figure 15 where we move from trajectories consisting of irregularly batched peaks (e.g. Figure 15(b)) towards trajectories consisting of more evenly distributed peaks (e.g. Figure 15(d)).

3.2.4. On the convergence of estimates in function of the observation length

We stress that the HASL based measures of oscillations discussed in this paper are, inherently, approximations of the actual quantities (e.g. average cycle duration and fluctuation) of a stochastic oscillator model. This is due to the fact that HASL is a statistical, rather than a numerical, model checking approach,



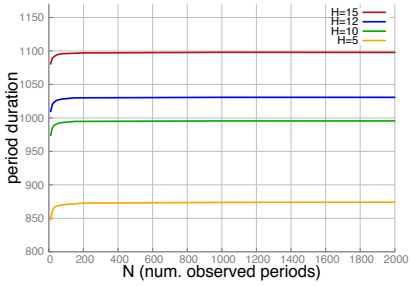
(a) $d' \in [0.01, 0.3]$



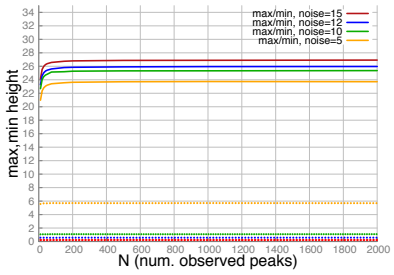
(b) detail for $d' \in [0.01, 0.25]$

Figure 16: Average period duration versus period fluctuation for the repressilator with 33x faster expression measured in function of degradation rate d'.

therefore all quantities are calculated with respect to (sampled) trajectories of finite length (i.e. either time-bounded or event-bounded trajectories). In order to assess how fast the measured quantities (cycle durations and peaks of oscillation) converge for the repressilator model we performed a number of re-iterated experiments where we evaluated the same quantity but changed the length of each observation (i.e. the number of observed cycles or peaks in each trajectory). Figure 17 depicts results of repeated measurement of the



(a) convergence of mean cycle duration



(b) convergence of min/max peaks mean value

Figure 17: Convergence of the mean value of the cycle duration and of the maximal/minimal peaks for the repressilator and with respect to the number of observed cycles/peaks

cycle duration (Figure 17(a)) and maximal/minimal peaks (Figure 17(b)) in function of N, the number of observed cycles, resp. peaks. Each graph contains different plots corresponding to different settings of the relevant parameters of the experiments (i.e. the L, H thresholds for the cycle measures of Figure 17(a) and the noise level for the peaks measures of Figure 17(b)). The obtained results indicate that, both for the cycles and peaks measures, a stable value is reached at

approximately $N \geq 200$. Based on this all (the above discussed) experiments for analyzing the oscillatory dynamics of the repressilator model have been executed by considering $N \geq 200$ observations on each trajectory. Finally observe that Figure 17 also illustrates how the measured quantity depends on the parameters of the measuring setting (i.e. the L, H thresholds of $\mathcal{A}_{per}$ and the noise level δ of $\mathcal{A}_{peaks}$). For the cycle duration we have repeatedly measured the duration by varying the H threshold parameter ($H \in \{5, 10, 12, 15\}$) of $\mathcal{A}_{per}$ while keeping the low threshold constant $L = 1$. Results in Figure 17(a) show that the measured cycle duration increases as we increase H. This result is not surprising as, by increasing H (while keeping L constant), we essentially instruct our cycle detector to exclude from detection those maximal peaks of lower height, hence, unless we change the stochastic characteristics of the oscillator, this necessarily implies that it will take longer for a trajectory to reach a higher H crossing point. Similarly Figure 17(b) reports on re-iterated measures of the min/max peaks obtained by using different values of the noise parameter δ of $\mathcal{A}_{peaks}$. Observe that in Figure 17(b) the values of maximal peaks are represented by full lines (each color corresponding to a different value for the noise parameter δ) while that of corresponding minimal peaks by a dashed line of the same color. Concerning the minimal peaks, the mean value is between 0 and 1 for noise 10 to 15 whereas it jumps to 6 for noise level 5. This suggests that a noise level of 5 is too small, hence detecting spurious minimal peaks.

3.2.5. Related work

The application of temporal logic reasoning to the analysis of oscillatory trends of stochastic models has been considered before. CSL-based characterizations of oscillations for CTMC models of biochemical reactions have been considered, with limited success, in [12] and more comprehensively in [53, 52]. In particular in [53] it has been shown that oscillations can be detected with CSL through an automata-based procedure. This, however, requires a manual hard-wiring of the detector-automaton within the CTMC model. The procedure has been demonstrated by implementation of examples on the PRISM model-checker. The main drawback of such an approach is indeed due to the required hard-wiring of the automaton within the CTMC model, a rather costly practice from the modeling point of view, which provides little flexibility. Recently, measuring of oscillations have been considered with other statistical model checking tools, i.e. UPPAAL-SMC [57], using MITL [23] and PLASMA [36], using BLTL [29]. However, the approach used for detection of the cycle duration in that case is inherently limited: i.e. the level of nesting of a MITL formula for detecting multiple occurrences of cycles is proportional to the number of cycles one wants to detect. More generally, the approaches in [12, 23] look at a simplification of the much more generic problem of oscillation detection whose complexity has been nicely illustrated in [52]. Another interesting contribution in the field of analysis of oscillations is presented in [37], where the relationship between stochastic oscillators and their continuous-deterministic counterpart is analyzed, however by mathematical approaches rather than temporal logic based ones.

4. Analysis of Single Gene model with stochastic delayed dynamics

We illustrate the application of HASL model checking to the analysis of a model of gene expression with stochastic delayed dynamics. We (1) present the NMSPN encoding of the considered model; (2) introduce a number of relevant properties/measures of a model, first describing them informally and then providing their encoding in HASL terms; (3) discuss results obtained by evaluation of the presented properties/measures by means of the COSMOS model-checker.

4.1. Stochastic models of gene expression with delayed dynamics

Gene expression is the process by which proteins are synthesized from a sequence in the DNA. It consists of two main phases: *transcription* and *translation*. Transcription is the copying of a sequence in the DNA strand by an RNA-polymerase (RNAp) into an RNA molecule. This process consists of three main stages: initiation, elongation and termination. Initiation consists of the binding of the RNAp to a promoter (Pro) region, unwinding the DNA and promoter escape. Afterwards, elongation takes place, during which the RNA sequence is formed, following the DNA code. Once the termination sequence is reached, both the RNAp and the RNA are released. In prokaryotes, translation, the process by which proteins are synthesized from the (transcribed) RNA sequence, can start as soon as the Ribosome Binding Site (RBS) region of the RNA is formed. The rate of expression of a gene is usually regulated at the stage of transcription, by activator/repressor molecules that can bind to the operator sites (generally located at the promoter region of the gene) and then promote/inhibit transcription initiation. Evidence suggests that this is a highly stochastic process (see, e.g. [5]) since, usually, the number of molecules involved, e.g. transcription factors and promoter regions, is very small, ranging from one to a few at a given moment [56]. Consequently, stochastic modeling approaches were found to be more appropriate than other strategies (e.g. ODE models or Boolean logic [55]).

The first stochastic models of gene expression assumed that the process of gene expression, once initialized, is instantaneous [5]. Namely, each step was modeled as a uni- or bi-molecular reaction and its kinetics was driven by the stochastic simulation algorithm (SSA) [30]. These models do not account for one important aspect of the kinetics of gene expression, namely that it consists, as mentioned, of a sequential process whose intermediate steps, once initiated, take considerable time to be completed (see e.g. [38]). This feature can be accounted for by introducing 'time delays' in the appearance of the products modeling the process [18, 50, 46]. To cope with this necessity, biochemical reactions with stochastic delays were introduced, represented in the following form:

$$R : \sum n_i React_i \xrightarrow{k} \sum m_j Prod_j^{nd} + \sum m'_h Prod_h^{d}(dist_h)$$

where $React_i$, $Prod_j^{nd}$ and $Prod_h^{d}$ denote respectively the i-th reactant, the j-th non-delayed product and the h-th delayed product ($n_i, m_j, m'_h \in \mathbb{N}$ being

the stoichiometric coefficients) and $dist_h$ denotes the distribution for the delayed introduction of h-th delayed product. Thus, for example, a reaction like $A + B \xrightarrow{k} A + C(\delta(\tau))$ represents an interaction between a molecule of A and one of B, that results in the transformation of the reacting B molecule into a C molecule (while the A molecule is left unchanged). The beginning of such interaction is assumed to occur after an exponentially distributed delay proportional to kinetic rate constant k. Once A and B interaction begins, the reacting A molecule is immediately released whereas the produced C molecule will appear after τ time units (i.e. $\delta(t)$ denotes a deterministic distribution with duration t). To deal with delayed reactions different adaptations of Gillespie's Stochastic Simulation Algorithm (referred to as "delayed SSA") have been introduced (e.g. [18, 15, 50, 18, 46]) yielding to the realisation of dedicated software tools such as SGNSim [48].

4.2. Single gene expression model

We consider a model of single gene expression that follows the approach proposed in [46]. Our model differs in that transcription is modeled as a 2-step process so as to accurately account for the open complex formation and promoter escape [38]. Each of these processes duration follows an exponential distribution, hence each reaction occurs with an exponentially distributed delay proportional to her kinetic rate (i.e. k_t for R_1, λ_1 for R_2, etc.). The gene expression system we refer to consists of the following reactions[9]:

$$R_1 \quad : \quad Pro + *RNAp \xrightarrow{k_t} Prox \tag{6}$$

$$R_2 \quad : \quad Prox \xrightarrow{\lambda_1} Pro + RBS + RNA(\Gamma(G_{len}, 0.09)) \tag{7}$$

$$R_3 \quad : \quad *Rib + RBS \xrightarrow{k_{tr}} RBS(\delta(\tau_1)) + Rib(\Gamma(G_{len}, 0.06)) + P(\Gamma(\tau_{5_{sh}}, \tau_{5_{sc}})) \tag{8}$$

$$R_4 \quad : \quad RBS \xrightarrow{rbsd} \emptyset \tag{9}$$

$$R_5 \quad : \quad Pro + Rep \xrightarrow{k_r} ProRep \tag{10}$$

$$R_6 \quad : \quad ProRep \xrightarrow{k_{unr}} Pro + Rep \tag{11}$$

Reactions (6) and (7) model transcription. In (6), an RNAp binds to a promoter (Pro), which remains unavailable for more reactions until reaction (7) occurs. Following reaction (7), which models the promoter escape, both the promoter and the RBS become immediately available at completion of R_2 whereas a complete RNA is released in the system after a further delay established according to a Gamma distribution $\Gamma(G_{len}, 0.09)$ whose parameter G_{len} is determined by the length of the gene, here set to 1000 nucleotides, and the

[9] note that symbol $*$ prefixing a species name in the above reactions means that the reactant is not consumed in the reaction. This is applied for simplicity to those reactants such as ribosomes, which exists in large amounts, and thus fluctuations in their numbers will not be significant in the propensity of reactions.

time spent by the RNAp at each nucleotide, which follows an exponential distribution with a mean of 0.09s [44]. Furthermore the kinetic rates k_t and λ_1 of reactions (6) and (7) are set to 1/400s, following measurements for the lar promoter in charge of transcription in Escherichia Coli [38]. Notice that, in this model, RNA will not be substrate to any reaction, and is only modeled as a means to count the number of RNA molecules produced over a certain period of time. In our model, according to the delayed SSA, the time necessary for any reaction to occur follows an exponential distribution whose mean is determined by the product between the rate constant of the reaction with the number of each of the reacting molecules present in the system at that moment.

In Prokaryotes, translation can begin as soon as the ribosome binding site (RBS) region of the RNA is completed. In reaction (8), a ribosome (Rib) binds to the RBS and translates the RNA. The RBS becomes available for more reactions after τ_1 s. The ribosome is released after $\Gamma(G_{len}, 0.06)$ seconds. The initiation rate, k_{tr} is set to 0.00042 s^{-1} [59]. Following measurements from *E. coli*, we have set $\tau_3 = 2$ s, and $\Gamma(G_{len}, 0.06)$ to follow a gamma distribution dependent on the gene's length, where each codon is added following an exponential distribution with a mean of 0.06 s [44]. Finally, $\Gamma(\tau_{5_{sh}}, \tau_{5_{sc}})$ is such that it accounts for the time that translation elongation takes, as well as the time it takes for a protein to fold and become active. In this case, we used the parameter values measured from GFP mutants commonly used to measure gene expression in *E. coli* [45]. Finally, we consider also three additional reactions representing respectively: RBS decay (9), promoter repression (10) and its reverse (11). Initially, the system has 1 promoter and 100 ribosomes. In the remainder of the paper we illustrate a thorough analysis of the above described single gene model by means of HASL model checking.

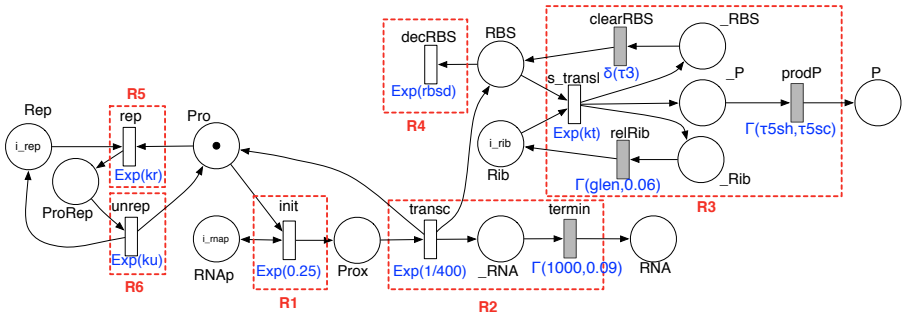


Figure 18: NMSPN model of Single Gene system with delayed stochastic dynamics

The single-gene model described by equations (6) to (11) is encoded as an NMSPN by the net depicted in Figure 18. The net includes a place for each species of the model (i.e. Pro, RNAp, Prox, RNA, RBS, Rib, P, Rep and ProRep) plus a number of auxiliary places representing intermediate stages of delayed reactions (i.e. _RNA, _RBS, _P, _Rib). Initial marking of the net is set by means of parameters *i_rep*, *i_rnap*, *i_rib*, which correspond to the chosen

initial population of the model (note that the promoter place *Pro* is initialized with one token, as each gene has one promoter region).

Reactions $R1$ to $R6$ of the single gene model correspond to subnets (enclosed in dashed rectangles) in Figure 18. Each such subnet contains either a single exponentially-distributed transition (in case of reactions with non-delayed products i.e. R1, R4, R5, R6) or a combination of exponential and non-exponential transitions (in case of reactions with delayed products, i.e. R2 and R3). For example, subnet R3 in Figure 18 represents the encoding of the *translation* reaction. It consists of: the translation-start event (i.e. exponentially distributed transition labeled *s_transl*, the RBS release event (i.e. deterministically distributed transition *clearRBS*), the ribosome releasing event (gamma distributed transition *relRib*) and the protein production event (i.e. gamma distributed transition *prodP*). Observe that the effect of repressed gene-expression can be promptly analyzed by setting up the repressor's initial population (parameter *i_rep*$= l_r$): unrepressed configurations corresponds to $l_r = 0$, whereas $l_r > 0$ settings correspond to repressed model where the level of repression is proportional to $l_r > 0$.

performance of TRANSCRIPTION and TRANSLATION mechanisms	
ID	**description**
ϕ_{1_a}	expected num. of completed-transcriptions (within T)
ϕ_{1_b}	expected num. of completed-translations (within T)
ϕ_{2_a}	prob. density of the number of completed-transcriptions (within T)
ϕ_{2_b}	prob. density of the number of completed-translations (within T)
ϕ_{3_a}	cumulative prob. of the number of completed-transcriptions (within T)
ϕ_{3_b}	cumulative prob. of the number of completed-translations (within T)
efficiency of TRANSLATION wrt TRANSCRIPTION	
ID	**description**
ϕ_4	exp. num. of completed translations between two consecutive transcriptions
ϕ_5	prob. of at least N completed translations between two consecutive transcriptions
REPRESSION related measures	
ϕ_6	percentage of time a gene is repressed
ϕ_7	how long does it take for translation to stop once a repression starts (i.e. sustainment of translation under repression)

Table 3: Properties of the Single Gene model

4.3. Properties of single gene model

Table 3 depicts an excerpt of (informally stated) relevant measures of the single-gene model. They are grouped according to different aspects of gene-expression performance. The corresponding HASL encoding is given in Table 4. We briefly illustrate the automata of Table 4 and the associated HASL expressions:

$\boldsymbol{\mathcal{A}_1}$ is designed for measures concerning the occurrences of *transc* and *transl* events. It accepts all paths of duration T and uses variables, n_1 and n_2 to maintain the number of *transc* and *transl* transitions occurring along a path. Different measures can be assessed through different HASL expressions referred to $\mathcal{A}_1$ including: $\phi_{1_a} = (\mathcal{A}_1, E[last(n_1)])$; $\phi_{1_b} = (\mathcal{A}_1, E[last(n_2)])$ and $\phi_4 =$

$(\mathcal{A}_1, E[last(n_2)/last(n_1)])$ (see Table 3) and their PDF and CDF equivalent ϕ_{2_a} to ϕ_{3_b}.

$\boldsymbol{\mathcal{A}_2}$ accepts only paths along which the number of transcriptions is $n_1 = C$ (within time T). The probability to have precisely C transcriptions within time T can be computed through specification $\phi_{2_a} = (\mathcal{A}_2, P)$.

$\boldsymbol{\mathcal{A}_3}$ is designed for measures concerning the amount of time a gene is repressed. Apart from the usual global clock t it uses a timer t_r registering the time gene is repressed, hence it grows ($\dot{t_r} = 1$) only in location l_1 (i.e. repression is ON, corresponding to a marking of place $ProRep > 0$), while it remains constant ($\dot{t_r} = 0$) in location l_0 (i.e. marking of place $ProRep = 0$). Also note that both l_0 and l_1 are initial locations, which is perfectly legal as their constraints make them mutually exclusive (this way $\mathcal{A}_3$ can be used to analyze both *repressed* and *unrepressed* configurations of the model).

$\boldsymbol{\mathcal{A}_4}$ measures *"how likely it is that, within a transcription interval (i.e. the interval between two occurrences of the* transc *event), at least N translations have been completed"*. It uses variables n_1 and n_2 (as above) and n_3 to count how many transcription intervals (along a path) contain $n_2 \geq N$ translations. The result is stored in $p_1 = n_3/n_1$ on acceptance. Note that, in this case, we consider an event-bounded observation window consisting of $n1 = N_1$ transcription events. Measure ϕ_5 of Table 3 in HASL terms is $\phi_5 = (\mathcal{A}_4, E[last(p_1)])$.

$\boldsymbol{\mathcal{A}_5}$ is designed for measures of *sustainment of translation activity under repression* (i.e. ϕ_7 in Table 3). It uses the following variables: n_o counting the number of repression intervals (interval between two repression events) in which translation arrested; t_o: measuring the *translation time-to-arrest* in a repression interval (given that translation arrested); T_o timer measuring the cumulated t_o. Note that translation arrest corresponds to the absence of tokens in all translation related places of the NMSPN model (Figure 18), corresponding to condition: $(RBS = 0 \wedge \_RBS = 0 \wedge \_P = 0 \wedge \_Rib = 0)$. Locations l_0, l_1 and l_2 are then associated to the following state conditions of the model: *repression is off* (l_0), *repression is on and translation off* (l_1) and *repression is on and translation ongoing* (l_2). All paths of duration $t = T$ are accepted and the target measure[10] is be obtained through expression $Z = E[last(T_o)/last(n_o)]$.

4.4. Experiments.

We assessed the previously described HASL measures through experiments executed with the COSMOS model-checker. For time-bounded measures we have considered (following [48]) $T = 2 \cdot 10^5$ as time horizon which roughly corresponds to 60 cell cycles, considering an average cell cycle of about 55 minutes (i.e. 3300s) in the case of E. coli. All experiments have been run with the following

[10] note that with a time-bounded measurement, as with $\mathcal{A}_5$, measuring may stop in any instant (not necessarily at the end) of a repression interval: this is not a problem as T_o and n_o are updated only when translation arrests, thus if bound T is reached before translation arrests, measure T_o/n_o will correctly refer to the duration of translation sustainment over all completed repression cycles

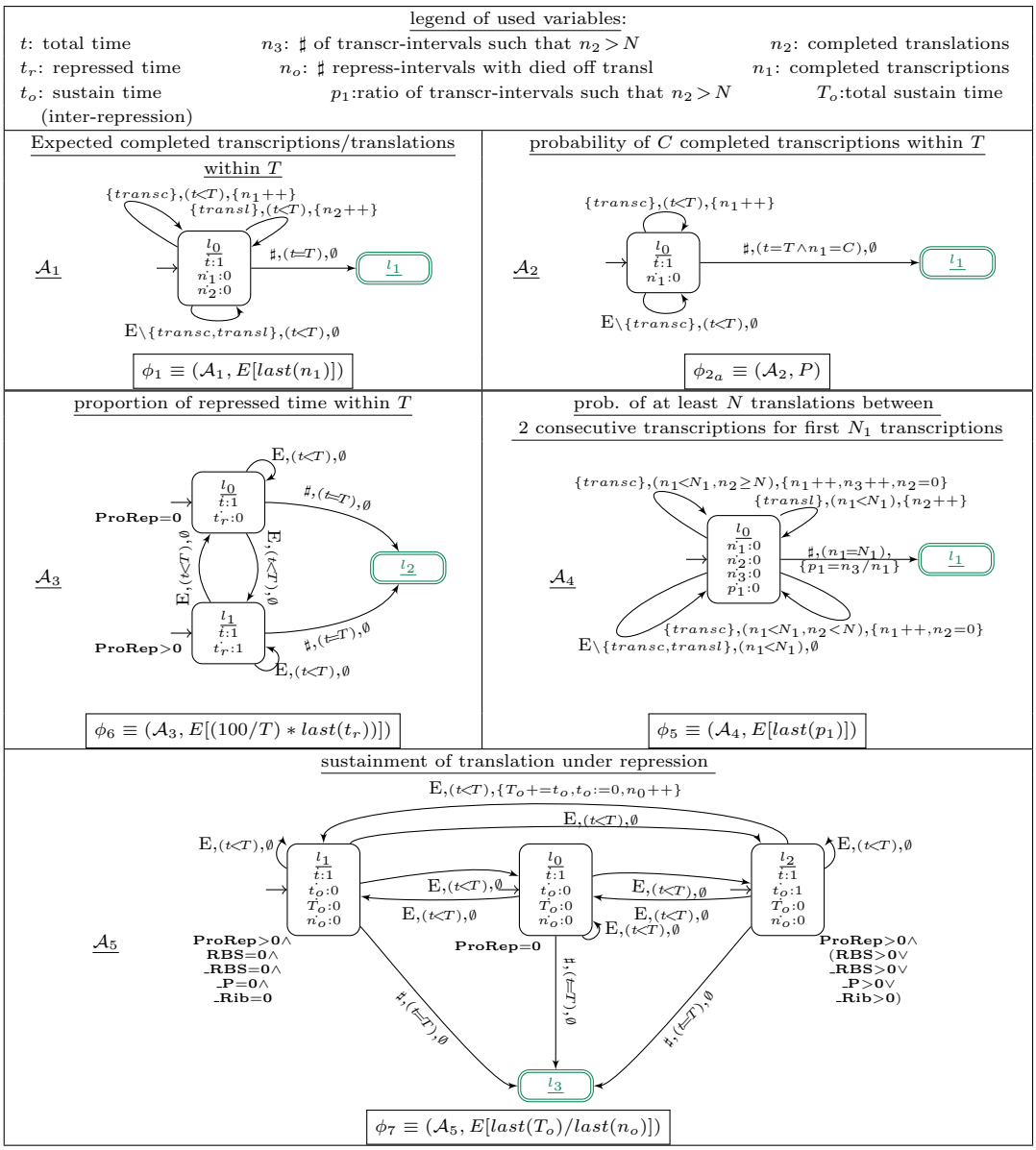
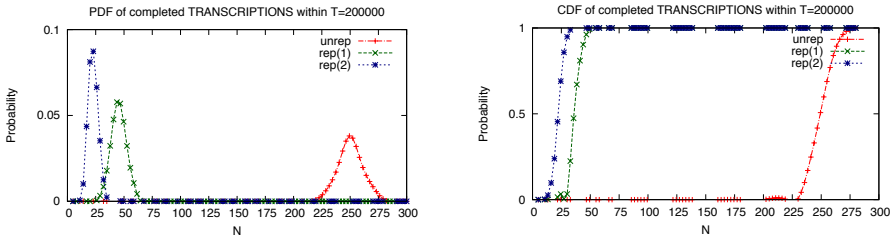


Table 4: LHA for various measures of the Single Gene model

settings concerning confidence interval estimation: confidence-level: 99.99%; interval-width: 0.01.

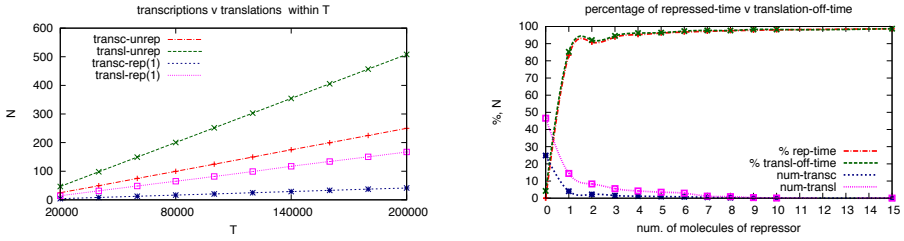


(a) PDF (b) CDF

Figure 19: PDF and CDF of completed transcriptions within T

Experiment 1. Figure 19 compares plots of the PDF (Figure 19(a)) and CDF (Figure 19(b)) of random variable n_1: *num. of completed transcription within* T (query ϕ_2 and ϕ_3) of *unrepressed* vs *repressed* configurations (i.e. *rep(1)*, corresponding to initial marking $i\_rep = 1$ and *rep(2)*, corresponding to initial marking $i\_rep = 2$). The effect of repression is evident as the bell-shaped probability density of n_1 is shifted toward lower values when increasing the level of repression.

Experiment 2. Figure 20(a) compares the *expected number of completed* transcriptions *vs.* translations *within* T in function of time for unrepressed and repressed (*rep(1)*) configurations. Observe that the throughput of *translation* is roughly twice as much as that of *transcriptions*, (both in unrepressed condition, as well as, in presence of repression). This is due to the rates of RNA degradation and translation initiation.



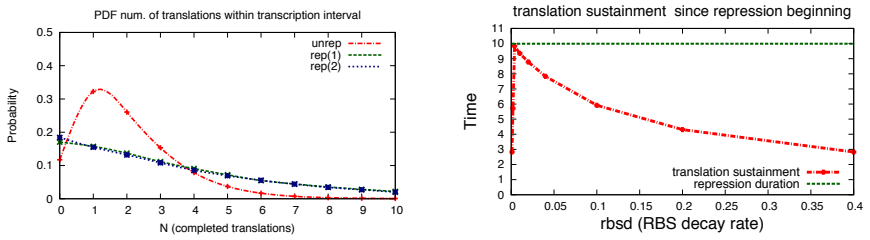
(a) unrep vs rep transc and transl (b) % rep-time v transc and transl

Figure 20: Exp. transcriptions and translations and percentage of repressed time

Experiment 3. Figure 20(b), plots two measures of timing: *the percentage of time gene is repressed* ($\mathcal{A}_3$) and the *percentage of time no translation activity is going on* (variant of $\mathcal{A}_3$) when system is observed for duration T and in

function of the level of repression (num. of repressor molecules on the x-axis). To observe also the trend of transcription and translation activity in function of repression level, Figure 20(b) also includes two curves referred to the expected number of *transcriptions*, respectively *translations*, within T. Observe that the presence of a single repressor is sufficient for the gene to remain repressed for 83% of the time and, likewise, for translation to be non-existing for 85% of the observation time (whereas in absence of repressor, translation activity is only non-existing for about 4% of the time).

Experiment 4. Figure 21(a) shows the PDFs of random variable n_2: *number of completed transcription within a transcription interval* (i.e. within two consecutive transcription completions) corresponding to query $\phi_4 : (\mathcal{A}_4, E[last(p_1)]$. This is computed for the unrepressed model and for two configurations of the repressed model (*rep(1)* and (*rep(2)*). Outcomes indicate that, in presence of repression, the probability density is more "distributed" than the bell shaped one corresponding to the unrepressed configuration. Furthermore, increasing the level of repression seems to have no great impact on the probability density (plots *rep(1)* and *rep(2)* are essentially identical).



(a) PDF num-transl in transcr-interval

(b) translation sustainment

Figure 21: Measure related to translation activity

Experiment 5. Figure 21(b) refers to measurement of the *translation sustainment within a repression-interval* (query $\phi_5 : (\mathcal{A}_5, E[last(n_{off})/last(n_{rep})])$ in function of the RBS decay rate (*rbsd*). We conducted our experiment with *rbsd* in the interval $[0.001, 4]$, which includes $rbsd = 0.01$ *i.e.* the value complying with experimental evidence used in the"standard" model's configuration. Obtained results indicate quite sensibly that translation sustainment is inversely proportional to RBS decay. It should be noted that with $rbsd < 0.004$ the translation sustainment is actually increasing with *rbsd* (not very evident in plot of Figure 21(b)). This is because, by definition, query $\phi_5 : (\mathcal{A}_5, E[last(n_{off})/last(n_{rep})])$ measures the sustainment of translation *on condition that sustainment lasts lesser than repression.* With $rbsd < 0.004$, however, decay is so slow that with high probability sustainment lasts longer than repression, while with low probability it lasts less. Thus it is sensible that the duration of *translation sustainment not exceeding repression duration* increases

for very small *rbsd*, i.e. $rbds \in [0, 0.004]$.

4.5. Discussion

We discuss here the contribution, in terms of the HASL based analysis of the gene-expression mechanism, considered in this section.

Some of the properties we have introduced here (Table 3) are just examples of typical performance measures (*e.g.* specification ϕ_1, corresponding to the average number of completed transcriptions/translations and its variants ϕ_2 and ϕ_3 which refer to the PDF, respectively CDF, of the average number of completed transcription/translation). ϕ_1 admits for an equivalent characterization through other stochastic logics, for example through the reward-enriched version of CSL [41] (supported by PRISM) which provide the user with dedicated reward-operators (e.g. $\mathcal{R}_{=?}$) and allows one to assess the average value of state and/or transition rewards cumulated along paths. On the other hand ϕ_2 and ϕ_3 refer to the PDF, respectively CDF, of a reward-based quantity (i.e. the average number of completed transcription/translation). The reward-based extension of PRISM CSL seems not to admit for a direct equivalent expression of measures like ϕ_2, ϕ_3 as in its syntax the argument φ of a probabilistic formula $\mathcal{P}_{=}?(\varphi)$ can only be a time-bounded path-formula (e.g. $\varphi \equiv (\Phi\ \mathcal{U}^{t\in I}\ \Psi)$ where $I \subseteq \mathbb{R}_{\geq 0}$ is the time-bounding interval) hence is not affected by reward-structures. On the contrary a logic like CSRL [20], which also extend CSL reasoning to CTMC enriched with state/impulse rewards, may admit an equivalent formulation for assessing the CDF of a reward-quantity (specifications like ϕ_3) as the argument φ of a CSRL probabilistic formula $\mathcal{P}_{=}?(\varphi)$ is a time/reward-bounded path formula (i.e $.(\Phi\ \mathcal{U}^{t\in I}_{r\in J}\ \Psi)$).

The remaining specifications $\phi_4, \phi_5, \phi_6, \phi_7$ (Table 3) are instead examples of properties that cannot be directly encoded through logics based on "simple" state/transition reward-structures like, for example, CSRL and the reward-extension of CSL featured by PRISM. This is because all of $\phi_4, \phi_5, \phi_6, \phi_7$ are based on rewards whose value must be updated in function of conditions observed along a path (e.g. with ϕ_4 we need to count the number of *translation* events occurred between two consecutive *transcription* events) and not only associated to the current state or to the transition being observed as it is the case with the reward-structures supported by CSRL and CSL.

The characterization of path-dependent reward-functions is normally achieved through association of an automaton (which selects the paths with relevant dynamics) to the input (CTMC) model. Thus, in principle, properties such as $\phi_4, \phi_5, \phi_6, \phi_7$ could still admit an equivalent formulation through logics such as CSRL and CSL at the cost of having, for each such property, to manually encode and incorporate an automaton into the considered input CTMC/MRM model.

To avoid such a costly burden, one can resort to automata-based logics (e.g. [25, 19, 11, 24]), i.e. logics that use a formalism using automata and allowing the user to explicitly characterize the dynamic behavior to be analyzed through an automaton. From the point of view of the expressiveness these logics are similar. However, differences exist depending on the class of automata

a logic is based on. For example with timed-automata (TA) logics, such as CSL^{TA} [25] or its n-clock extension [19], one can assess the probability that the considered model exhibits sophisticated timing conditions. However, in their original formulation these logics do not support reward-based analysis hence they are constrained to the evaluation of probability measures. Thus, even if semantically the paths associated to timing-dependent properties such as $\phi_6 \equiv (\mathcal{A}_3, E[(100/T) * last(t_r))])$ and $\phi_7 \equiv (\mathcal{A}_5, E[last(T_o)/last(n_o)]$ could be characterized through CSL^{TA} and its [19] extension, the actual target measures cannot since both $E[(100/T) * last(t_r))]$ and $E[last(T_o)/last(n_o)]$ are in fact reward measures. Moreover, the target measure $E[last(T_o)/last(n_o)]$ depends of an integer variable (n_0, the number of observed *transcription* intervals in which *translation* has completed), hence could not be expressed anyway by any logic based on a "pure" TA formalism as TA are limited to clock variables (i.e. variables that can increase with gradient 1 or stay constant and whose update can only be a reset).

Similar considerations apply also to specifications ϕ_4 and ϕ_5 which represent a reward-measure (ϕ_4), respectively a probability measure (ϕ_5) based on counting certain events within an observed event pattern. On the other hand, logics based on generalized version of TA, such as Linear Hybrid Automata [11] or Priced Timed Automata [24], avoid the limitations of TA logics by allowing the modeler to use/combine any type of variables (real, integer, boolean) in order to express the relevant characteristic of the paths to observe as well as the target measure (probability of reward) to be evaluated.

Finally, we stress that the increased expressiveness achieved through automata-based specifications comes at the cost of having to get acquainted with *expressing a measure through an automata*. This is the main drawback of automata-based specification formalisms. Within HASL the development of a higher level specification language (e.g. based on some sort of temporal-logic-like syntax), suitable for automatic translation into a corresponding automata specification is certainly conceivable and part of the future work, as discussed in [11].

5. Conclusion

In this paper we have considered the application of an expressive stochastic specification language, namely HASL, to the analysis of biological models. First we have considered the problem of analyzing oscillations, and in that respect we have presented HASL specifications for assessing both *noisy periodicity* and *noisy alternance*. We have applied this approach on a real biological oscillator, evaluating expected values of interesting measures, as well as their PDF and CDF. In the second part of the paper we focused on another type of problem, namely the analysis of gene-expression with stochastic delayed dynamics. This type of models often involve non-exponential delay distribution, and thus cannot be analyzed using numerical stochastic model checking approaches. In the analysis of gene-expression, we demonstrated the potential of HASL by illustrating examples of sophisticated analysis, such as the measurement of the efficiency of the transcription-phase with respect to the translation-phase of the

expression process (i.e. by means of measures of the sustainment of translation with respect to transition).

Acknowledgments

The authors would like to warmly thank Andre S. Ribeiro, Jarno Mäkelä and Ilya Popatov for their precious contribution to this work, particularly with respect to the part regarding modeling of gene-expression with stochastic delayed dynamics.

References

[1] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley & Sons, 1995.

[2] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, LNCS 736, 1992.

[3] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2), 1994.

[4] E. G. Amparore, P. Ballarini, M. Beccuti, S. Donatelli, and G. Franceschinis. Expressing and computing passage time measures of gspn models with hasl. In J. M. Colom and J. Desel, editors, *Petri Nets*, volume 7927 of *Lecture Notes in Computer Science*, pages 110–129. Springer, 2013.

[5] A. P. Arkin, J. Ross, and H. H. McAdams. Stochastic Kinetic Analysis of a Developmental Pathway Bifurcation in Phage-l Escherichia coli. *Genetics*, 149(4):1633–1648, 1998.

[6] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Model-checking CTMCs. *ACM Trans. on Computational Logic*, 1(1), 2000.

[7] C. Baier, L. Cloth, B. Haverkort, M. Kuntz, and M. Siegle. Model checking action- and state-labelled Markov chains. *IEEE Trans. on Software Eng.*, 33(4), 2007.

[8] C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Model-checking algorithms for CTMCs. *IEEE Trans. on Software Eng.*, 29(6), 2003.

[9] C. Baier and J.-P. Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[10] P. Ballarini, H. Djafri, M. Duflot, S. Haddad, and N. Pekergin. COSMOS: a statistical model checker for the hybrid automata stochastic logic. In *Proceedings of the 8th International Conference on Quantitative Evaluation of Systems (QEST'11)*, pages 143–144. IEEE Computer Society Press, sep. 2011.

[11] P. Ballarini, H. Djafri, M. Duflot, S. Haddad, and N. Pekergin. HASL: an expressive language for statistical verification of stochastic models. In *Proc. Valuetools*, 2011.

[12] P. Ballarini and M. Guerriero. Query-based verification of qualitative trends and oscillations in biochemical systems. *Theoretical Computer Science*, 411(20):2019 – 2036, 2010.

[13] P. Ballarini, J. Mäkelä, and A. S. Ribeiro. Expressive statistical model checking of genetic networks with delayed stochastic dynamics. In *CMSB*, pages 29–48, 2012.

[14] R. Barbuti, F. Levi, P. Milazzo, and G. Scatena. Probabilistic model checking of biological systems with uncertain kinetic rates. *Theor. Comput. Sci.*, 419:2–16, Feb. 2012.

[15] M. Barrio, K. Burrage, A. Leier, and T. Tian. Oscillatory regulation of hes1: Discrete stochastic delay modelling and simulation. *PLoS Computational Biology*, 2(9):1017–1030, 2006.

[16] G. Bernot, J.-P. Comet, A. Richard, and J. Guespin. Application of formal methods to biological regulatory networks: extending Thomas' asynchronous logical approach with temporal logic. *Journal of Theoretical Biology*, 229(3):339–347, Aug. 2004.

[17] A. Bobbio, A. Puliafito, M. Telek, and K. S. Trivedi. Recent developments in non-Markovian stochastic Petri nets. *Journal of Circuits, Systems, and Computers*, 8(1):119–158, 1998.

[18] D. Bratsun, D. Volfson, L. S. Tsimring, and J. Hasty. Delay-induced stochastic oscillations in gene regulation. *Proc Natl Acad Sci U S A*, 102(41):14593–8, 2005.

[19] T. Chen, T. Han, J.-P. Katoen, and A. Mereacre. Quantitative model checking of CTMC against timed automata specifications. In *Proc. LICS'09*, 2009.

[20] L. Cloth, J.-P. Katoen, M. Khattri, and R. Pulungan. Model checking Markov reward models with impulse rewards. In *Proc. International Conference on Dependable Systems and Networks (DSN'05)*, pages 722–731. IEEE Computer Society Press, 2005.

[21] COSMOS home page. http://www.lsv.ens-cachan.fr/Software/cosmos/.

[22] A. David, K. G. Larsen, A. Legay, M. Mikucionis, and Z. Wang. Time for statistical model checking of real-time systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, pages 349–355, 2011.

[23] A. David, K. G. Larsen, A. Legay, M. Mikučionis, D. B. Poulsen, and S. Sedwards. Runtime verification of biological systems. In *Proceedings of the 5th international conference on Leveraging Applications of Formal Methods, Verification and Validation: technologies for mastering change - Volume Part I*, ISoLA'12, pages 388–404, Berlin, Heidelberg, 2012. Springer-Verlag.

[24] A. David, K. G. Larsen, A. Legay, M. Mikučionis, D. B. Poulsen, J. Van Vliet, and Z. Wang. Statistical model checking for networks of priced timed automata. In *Proceedings of the 9th International Conference on Formal Modeling and Analysis of Timed Systems*, FORMATS'11, pages 80–96, Berlin, Heidelberg, 2011. Springer-Verlag.

[25] S. Donatelli, S. Haddad, and J. Sproston. Model checking timed and stochastic properties with CSL^{TA}. *IEEE Trans. on Software Eng.*, 35, 2009.

[26] M. Eickhoff, D. McNickle, and K. Pawlikowski. Detecting the duration of initial transient in steady state simulation of arbitrary performance measures. In *Proceedings of the 2Nd International Conference on Performance Evaluation Methodologies and Tools*, ValueTools '07, pages 42:1–42:7, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[27] M. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(335), 2000.

[28] F. Fages, S. Soliman, and C. N. Rivier. Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM. *Journal of Biological Physics and Chemistry*, 4(2):64–73, 2004.

[29] B. Finkbeiner and H. Sipma. Checking finite traces using alternating automata. *Formal Methods in System Design*, 24(2):101–127, 2004.

[30] D. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[31] P. W. Glynn. A gsmp formalism for discrete event systems. *Proceedings of the IEEE*, 77, 1989.

[32] R. Gorrieri, U. Herzog, and J. Hillston. Unified specification and performance evaluation using stochastic process algebras. *Perform. Eval.*, 50(2/3):79–82, 2002.

[33] B. Haverkort, L. Cloth, H. Hermanns, J.-P. Katoen, and C. Baier. Model checking performability properties. In *Proc. DSN'02*, 2002.

[34] R. He, P. Jennings, S. Basu, A. P. Ghosh, and H. Wu. A bounded statistical approach for model checking of unbounded until properties. In *Proc. ASE'10*, 2010.

[35] J. Heath, M. Kwiatkowska, G. Norman, D. Parker, and O. Tymchyshyn. Probabilistic model checking of complex biological pathways. *Theoretical Computer Science*, 319(3):239–257, 2008.

[36] C. Jégourel, A. Legay, and S. Sedwards. A platform for high performance statistical model checking - plasma. In *Proc. 18th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'12)*, pages 498–503, 2012.

[37] J. Júlvez, M. Kwiatkowska, G. Norman, and D. Parker. Evaluation of sustained stochastic oscillations by means of a system of differential equations. *International Journal of Computers and Applications (IJCA)*, 19(2):101–111, 2012.

[38] M. Kandhavelu, A. Hakkinen, O. Yli-Harja, and A. S. Ribeiro. Single-molecule dynamics of transcription of the lar promoter. *Phys Biol*, 9(2), 2012.

[39] H. Kitano. *Foundations of Systems Biology*. MIT Press, 2002.

[40] D. E. Knuth. *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[41] M. Kwiatkowska, G. Norman, and D. Parker. Stochastic model checking. In *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation*, volume 4486 of *LNCS*, pages 220–270. Springer, 2007.

[42] M. Kwiatkowska, G. Norman, and D. Parker. Prism 4.0: Verification of probabilistic real-time systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, pages 585–591, 2011.

[43] M. Lakin, D. Parker, L. Cardelli, M. Kwiatkowska, and A. Phillips. Design and analysis of DNA strand displacement devices using probabilistic model checking. *Journal of the Royal Society Interface*, 2011. To appear.

[44] J. Makela, J. Lloyd-Price, O. Yli-Harja, and A. Ribeiro. Stochastic sequence-level model of coupled transcription and translation in prokaryotes. *BMC Bioinformatics*, 12(1):121, 2011.

[45] J. Megerle, G. Fritz, U. Gerland, K. Jung, and J. Rädler. Timing and Dynamics of Single Cell Gene Expression in the Arabinose Utilization System. *Biophysical Journal*, 95:2103–2115, 2008.

[46] A. Ribeiro, R. Zhu, and S. A. Kauffman. A general modeling strategy for gene regulatory networks with stochastic dynamics. *Journal of computational biology : a journal of computational molecular cell biology*, 13(9):1630–1639, Nov. 2006.

[47] A. Ribeiro, R. Zhu, and S. A. Kauffman. A general modeling strategy for gene regulatory networks with stochastic dynamics. *Journal of Computational Biology*, 13(9):1630–1639, 2006.

[48] A. S. Ribeiro and J. Lloyd-Price. SGN Sim, a stochastic genetic networks simulator. *Bioinformatics (Oxford, England)*, 23(6):777–779, Mar. 2007.

[49] S. M. Ross. *Simulation*. Academic Press, Elsevier, third edition edition, 2002.

[50] M. R. Roussel and R. Zhu. Validation of an algorithm for delay stochastic simulation of transcription and translation in prokaryotic gene expression. *Physical Biology*, 3(4):274–284, 2006.

[51] K. Sen, M. Viswanathan, and G. Agha. VESTA: A statistical model-checker and analyzer for probabilistic systems. In *Proc. QEST'05*, 2005.

[52] D. Spieler. Model checking of oscillatory and noisy periodic behavior in Markovian population models. Master's thesis, Saarland University, 2009.

[53] D. Spieler. Characterizing oscillatory and noisy periodic behavior in Markov population models. In *Proc. QEST'13*, 2013.

[54] W. Stewart. *Introduction to the numerical solution of Markov chains*. Princeton Univ. Press, Princeton, NJ, 1994.

[55] Z. Szallasi, J. Stelling, and V. Periwal. *System Modeling in Cellular Biology: From Concepts to Nuts and Bolts*. A Bradford book. Mit Press, 2010.

[56] Y. Taniguchi, P. J. Choi, G.-W. Li, H. Chen, M. Babu, J. Hearn, A. Emili, and X. Xie. Quantifying E. coli Proteome and Transcriptome with Single-Molecule Sensitivity in Single Cells. *Science*, 329(5991):533–538, July 2010.

[57] Uppaal-smc home page. `http://people.cs.aau.dk/~adavid/smc/`.

[58] H. Younes. Ymer: A statistical model checker. In *Proc. CAV'05*, LNCS 3576, 2005.

[59] R. Zhu, A. Ribeiro, D. Salahub, and S. Kauffman. Studying genetic regulatory networks at the molecular level: Delayed reaction stochastic models. *Journal of Theoretical Biology*, 246(4):725–745, 2007.

A. Appendix

We present in this appendix further details on the experiments using the HASL based approach to analyze stochastic oscillators. In Section A.1 we report on the analysis of the period duration and period fluctuation of a simple model of probabilistic square wave oscillator, and in Section A.2 we give details on how to reproduce the experiments on the repressilator model.

A.1. Checking the validity of the measures of period obtained through $\mathcal{A}_{per}$ on a simple example of probabilistic square wave oscillator

In order to validate the HASL based approach for oscillation related measures, we consider a simple example of probabilistic square wave oscillator, corresponding to the NMSPN model given in Figure 22. The interest in such a toy-example of synthetic oscillator is twofold. First its simplicity makes it possible to estimate measures (period, fluctuations) analytically and thus compare them with the measures obtained with HASL. Then its oscillatory characteristics (i.e. amplitude and period of oscillations) are well-defined and can be easily tuned through the model's parameters thus swapping between a deterministic oscillator (i.e. constant period) and a probabilistic one (period duration's associated to a discrete probability distribution).

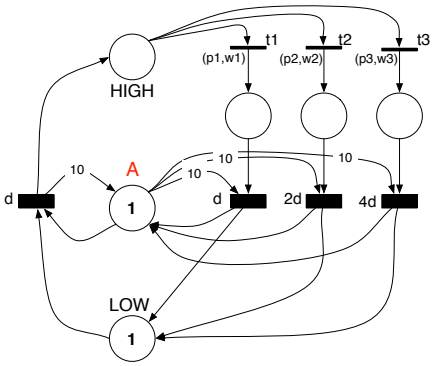


Figure 22: NMSPN model of a configurable probabilistic square wave oscillator

The model consists of a single biochemical species A whose population (corresponding to the marking of place A in Figure 22) periodically alternates between 1 molecule (low-level) and 10 molecules (high-level) through step-jump transitions. The duration of the low-level phase is fixed (d) and determined by the (deterministically distributed) timed-transition T_u, where d is a parameter of the NMSPN. On the other hand, each high-level phase may have any of 3 possible durations: $d, 2d$ or $4d$. The different durations available for the high-level phase are determined through the model's configuration, i.e through the chosen values for the *priorities* (π_1, π_2, π_3) and the *weights* (w_1, w_2, w_3) of the immediate transitions t_1, t_2 and t_3 (Table 5). According to the chosen configuration, we may distinguish between *regular square waves* (e.g. left-hand side in Figure 23), corresponding to waves of constant period (i.e. either $\Delta p = 2d$, $\Delta p = 3d$ or $\Delta p = 5d$), as opposed to *irregular square waves* (e.g. right-hand side in Figure 23), i.e. waves whose period vary probabilistically.

The NMSPN model of the square wave oscillator in Figure 22 consists of 3 major places: A (representing the population of species A) plus the two mutually-exclusive places HIGH and LOW (representing the current level of the wave). The timing of the wave (i.e. the period duration) is driven by

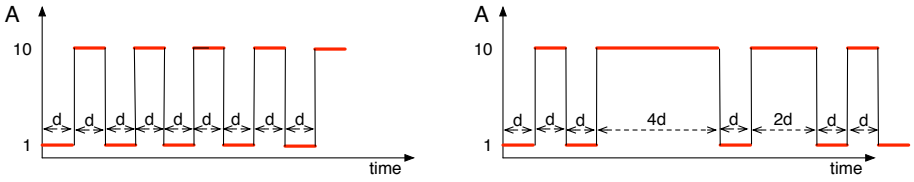


Figure 23: traces of a square wave oscillator with: (left) constant period ($\Delta p = 2d$); (right) probabilistically variable period ($\Delta p \in \{2d, 3d, 5d\}$)

Configurations of the square wave oscillator					
conf. name	**priorities** (π_1, π_2, π_3)	**weights** (w_1, w_2, w_3)	**nature**	**mean** μ_{t_p}	**fluctuation** f_{t_p}
regular_1	$(1, 0, 0)$	$(\_, \_, \_)$	det.	10	0
regular_2	$(0, 1, 0)$	$(\_, \_, \_)$	det.	15	0
regular_3	$(0, 0, 1)$	$(\_, \_, \_)$	det.	25	0
irregular_1	$(1, 1, 0)$	$(1, 1, \_)$	prob.	12.5	6.25
irregular_2	$(0, 1, 1)$	$(\_, 1, 1)$	prob.	20	25
irregular_3	$(1, 0, 1)$	$(1, \_, 1)$	prob.	17.5	56.25
irregular_4	$(1, 1, 1)$	$(1, 1, 1)$	prob.	16.67	38.89

Table 5: Configurations of the square wave oscillator: regular waves correspond to a single maximal priority (and in this case the weights are irrelevant); irregular waves correspond to at least two maximal priorities (and in this case the period duration is probabilistic and depends on the corresponding weights). The (exact) mean value (μ_{t_p}) and fluctuation (f_{t_p}) of the oscillation period computed for $d = 5$ are indicated in the last two columns.

the deterministic-timed transitions (represented as thick bars), each of which is associated with a specific delay-duration (i.e. a multiple of the parameter d). Finally, the possibility of switching between a regular square wave configuration and an irregular one is achieved by means of the parameters of the 3 immediate transitions t_1, t_2 and t_3 (represented as thin bars) which represent the selection of the duration of the high-level interval. Each such transition is associated with a *priority* and a *weight* ((π_i, w_i), $i \in \{1, 2, 3\}$) parameter. A transition with maximal priority is always chosen first. As a consequence a regular square wave is obtained by any (deterministic) configuration which guarantees that only one amongst t_1, t_2 and t_3 can be chosen (i.e. this is achieved by having a single maximum priority $\pi_i > max(\pi_j, \pi_k)$ $i, j, k \in \{1, 2, 3\}$ and $i \neq j \neq k$). Thus we have three possible regular square waves: $\text{reg}_1 : (\pi_1, \pi_2, \pi_3) = (1, 0, 0)$, $\text{reg}_2 : (\pi_1, \pi_2, \pi_3) = (0, 1, 0)$ and $\text{reg}_3 : (\pi_1, \pi_2, \pi_3) = (0, 0, 1)$

On the other hand, irregular square waves correspond to configurations where at least two transitions have equal maximum priority (i.e. $\pi_i = \pi_j > \pi_k$ or $\pi_i = \pi_j = \pi_k$). In this case the maximally prioritized transitions are probabilistically selected according to their respective weights. For example with priorities $(\pi_1, \pi_2, \pi_3) = (1, 1, 1)$ transition t_i is selected with probability $w_i / \sum w_j$. There are four possible irregular waves listed in Table 5. The mean value μ_{t_p} and the fluctuation of the period f_{t_p} for an irregular configuration can be straightfor-

wardly expressed analytically in function of the priority and weight parameters of transitions t_1, t_2, t_3. For example for the $\text{irr}_4 : (\pi_1, \pi_2, \pi_3) = (1, 1, 1)$ configuration they are given by (12), respectively (13).

$$\mu_{t_p} = \frac{w1}{\sum w_i} 2d + \frac{w2}{\sum w_i} 3d + \frac{w3}{\sum w_i} 5d \tag{12}$$

$$f_{t_p} = \frac{w1}{\sum w_i} (2d - \mu_{t_p})^2 + \frac{w2}{\sum w_i} (3d - \mu_{t_p})^2 + \frac{w3}{\sum w_i} (5d - \mu_{t_p})^2 \tag{13}$$

In the following experiments, we consider the configurations illustrated in Table 5. Note that, in the table, irregular configurations are defined so that all possible durations of the period are equally likely (i.e. the weights of maximally prioritized transitions have all the same value). This induces a (specific) discrete distribution of probability over the oscillation period t_p, for example with configuration *irregular1* (i.e. priorities $(1, 1, 0)$ weights: $(1, 1, 0)$): $Pr(t_p = 10) = Pr(t_p = 15) = 0.5$. As a consequence the exact mean value (μ_{t_p}) and fluctuation (f_{t_p}) of the period t_p can be straightforwardly obtained analytically: the actual values are indicated in Table 5. As expected, deterministic configurations have no fluctuation. On the other hand the most irregular waves (i.e. those whose period varies the most) correspond to priorities configuration $(1, 0, 1)$: such waves will alternate between equally likely high-level phases of duration d and others of duration $4d$.

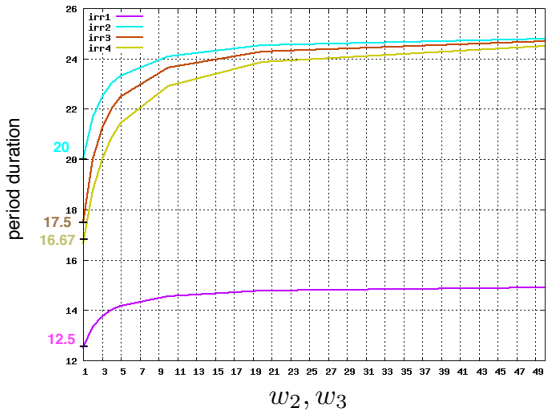


Figure 24: Measured average duration of the period for different configurations of the square wave oscillator, as a function of the weights, i.e. for irr_1 $w_2 \in [1, 50]$ and $w_1 = 1$, for irr_2 $w_3 \in [1, 50]$ and $w_2 = 1$, for irr_3 $w_3 \in [1, 50]$ and $w_1 = 1$ and for irr_4 $w_3 \in [1, 50]$ and $w_2 = w_1 = 1$.

Experiments on the square wave oscillator. Figure 24 and Figure 25 report on measurements of the average period, respectively the fluctuation of the period, for the square wave oscillators, obtained through the LHA for measuring noisy-periodic traces (i.e. Figure 7). For these experiments, knowing *a priori* the

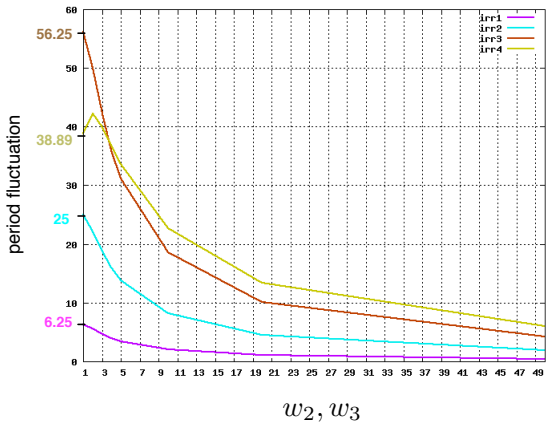


Figure 25: Measured fluctuation of the period duration for different configurations of the square wave oscillator, as a function of the weights i.e. for irr_1 $w_2 \in [1, 50]$, $w_1 = 1$, for irr_2 $w_3 \in [1, 50]$ and $w_2 = 1$, for irr_3 $w_3 \in [1, 50]$ and $w_1 = 1$ and for irr_4 $w_3 \in [1, 50]$ and $w_2 = w_1 = 1$.

amplitude of the square waves (i.e. min=1, max=10), we manually set the LHA thresholds to $L = 2$ and $H = 9$. The goal was then to assess measures related to the period. For the null-fluctuation (regular) configurations of the square wave oscillator (i.e. reg1, reg2 and reg3), the measured average duration and fluctuation of the period precisely match the theoretical ones depicted in Table 5. Figure 24 and 25, instead, refer to the irregular configurations. In particular the average value (respectively the fluctuation) of the period is measured as a function of either weight w_2 or w_3 (depending on the considered configuration), hence as a function of the probability of selecting either $2d$ or $4d$ as the duration of the high-level phase. For example the curves for configuration irr_2, irr_3 and irr_4 refer to measuring the effect on the average value and fluctuation of the period of the variation of $w_3 \in [1, 50]$ while keeping the w_1 and w_2 fixed. The curves for configuration irr_1 refer to measuring the effect of the variation of $w_2 \in [1, 50]$ while keeping $w_1 = 1$. For irr_2, we observe that augmenting w_3 increases the probability of observing a period of length $5d$, leading the oscillator to behave, for $w_3 \to \infty$, like the regular configuration reg_3. This is confirmed by the fact that the irr_2 curves in Figure 24 and 25 tends towards the $\mu_{t_p} = 25$, respectively $s^2_{t_p} = 0$ values of the regular configuration reg_2. Notice that for all plots in Figure 24 and 25 the measured value on $x = 1$ matches the exact value of the corresponding irregular configuration (see Table 5), which is a good indication of the accuracy of the HASL measure. For the measure period fluctuation Figure 24 notice that the decreasing shape of each curve confirm the intuition: the irregularity of the period (i.e. its fluctuation) depends on the distribution of probability between the possible different durations. Intuitively the more uniform such probability is distributed amongst several possible durations the higher the fluctuation of the period. By contrast at the limit $w_2 \to \infty$ (or

Experiments for measuring the PDF of maximal peaks through $\mathcal{A}_{peaks}$					
noise	**tot. paths**	**interval width**	**sim. runtime (s)**	**♯jobs**	**build time (s)**
1	23000	0.000224	109.08	4	1.12
2	15800	0.000326	166.91	4	1.44
3	15650	0.000330	282.13	4	1.34
5	15250	0.000339	466.35	4	1.25
10	15050	0.000344	750.84	4	1.10
15	16400	0.000316	969.44	4	1.37
20	18800	0.000302	1388.56	4	1.40
25	25150	0.000209	2753.07	4	1.15

Table 6: Data concerning experiments run with $\mathcal{A}_{peaks}$ for measuring the PDF of the maximal peaks of oscillations for the repressilator model

$w_3 \to \infty$) the fluctuation asymptotically tends to zero. In a similar fashion, we also validated the LHA for assessing noisy-periodically alternating traces. In particular with the LHA in Figure 5 we measured the amplitude and the period of the square wave oscillator obtaining the correct results (to save space we omit to report them here).

A.2. Experiments for measuring the oscillation of the Repressilator

In the following we provide some information concerning the experiments for the analysis of the repressilator model described in Section 3.2.1. All these experiments have been done with COSMOS and using the default configuration for what concerns the confidence-level (i.e. 0.99) and the interval-width (i.e. 0.001). Table 6 refers to experiments for the measure of the PDF of maximal peaks of oscillations obtained through $\mathcal{A}_{peaks}$ (see Figure 11).

It shows various relevant information for each experiment performed through $\mathcal{A}_{peaks}$. Specifically for each experiment Table 6 illustrates: the different values of $\mathcal{A}_{peaks}$'s *noise* parameter used for the experiment (first column); the number of generated path (second column); the actual width of the confidence interval at the end of the estimation (third column); the actual (total) simulation for completing the experiment (fourth column); the number of jobs executed in parallel[11] (fifth column); the build time for compiling the simulator (sixth column). COSMOS employs a code-generation scheme through which a customized simulator is built for every single experiment by compilation of the GSPN model and of the LHA specification (see [10]). The total run time of an experiment is then given by the sum of the simulation time (fourth column) and the time to build (compile) the simulator for the experiment (sixth column). The interested reader can reproduce the experiments discussed in this paper by retrieving the GSPN models and HASL specifications files available at COSMOS web-page `http://www.lsv.ens-cachan.fr/Software/cosmos/`. Instructions

[11] i.e. with COSMOS the user can choose to split the simulation of batches of traces in parallel on a given number of jobs which will be then mapped by the OS on the actual cores of the CPU.

on how to install and run COSMOS are available in the archive containing the entire COSMOS distribution (available on the COSMOS web-page).